

Liste d'exercices

Document rédigé par Raphaël
Révision par André, Martin et Rafik

Table des matières

1	Syntaxe, variables et I/O	4
1.1	Explications 1	4
1.2	Erreurs 1	4
1.3	Complétions 1	5
1.4	Alice et Bob 1	6
1.5	Triple d'un nombre	6
1.6	Aire d'un cercle	6
1.7	Une expression... compliquée!	7
1.8	Racines carrées	7
1.9	Puissances	7
1.10	Un peu de binaire	7
2	Structures conditionnelles et boucles	8
2.1	Explications 2	8
2.2	Erreurs 2	9
2.3	Complétions 2	10
2.4	Du tableau à la boucle	12
2.5	Parité	12
2.6	Les premiers nombres premiers	12
2.7	Stan Robotix 1	12
2.8	La meilleure des structures conditionnelles	13
2.9	Nombres aléatoires 1	13
2.10	Nombres aléatoires 2	13
3	Vecteurs, tableaux et pointeurs	14
3.1	Tableau ou vecteur?	14
3.2	À couvert!	14
3.3	Une racine carrée... compliquée!	14
3.4	Prédictions 1	15
3.5	Alice et Bob 2	15
3.6	Un vecteur bien long	16
3.7	Stan Robotix 2	16
3.8	Céline et les statistiques	16
3.9	Un vecteur bien long... mais sans pairs!	16
4	Fonctions	18
4.1	Le meilleur des types de retour	18
4.2	Une fonction... spéciale!	18
4.3	Le plus petit des multiples	18
4.4	Mes carrés!	18
4.5	Céline et le jeu	18

4.6	In french please !	19
4.7	Prédictions 2	19
4.8	Nombres aléatoires 3	19
4.9	SohCahToa !	20
4.10	Centaines, dizaines et unités	20
4.11	Stan Robotix 3	20
5	Classes	22
5.1	Définitions	22
5.2	Objet construit !	22
5.3	Le retour de la racine carrée perdue	23
5.4	Triangle	23
5.5	Cercle	23
5.6	Le fameux hérisson	23
5.7	Une architecture classique	24
5.8	Jeu 1	24
5.9	Simulation	26
5.10	Prédictions 3	26
6	Git et Github	27
6.1	Des commandes et des explications	27
6.2	Entraînement	27
7	Exercices supplémentaires	28
7.1	Nombres aléatoires 4	28
7.2	Alice et Bob 3	28
7.3	Tri par sélection	29
7.4	Diviser pour mieux estimer	29
7.5	C'est plus ! C'est moins !	29
7.6	Alors c'est qui le plus malin ?	30
7.7	Stan Robotix 4	30
7.8	Genshin Impact et les chaînes de caractères	30
7.9	Stan Robotix et les chaînes de caractères	31
7.10	Neurone artificiel	32
7.11	En lettres s'il vous plait !	32
7.12	En chiffres s'il vous plait !	33
7.13	Chiffrement par décalage	33
7.14	Stan Robotix 5	33
7.15	Brute Force	33
7.16	Alice et Bob 4	33
7.17	La pyramide d'astérisques	34
7.18	Un grapheur de base	34
7.19	Marche aléatoire	34
7.20	Produits de vecteurs	35
7.21	Link, un archer hors-pair	35
7.22	Encore plus de binaire	35
7.23	Des chiffres mais pas trop	35
7.24	Un cercle et des polygones	36
7.25	Céline et les diviseurs	36
7.26	De l'imagination, il en faut !	36
7.27	Jeu 2	36
7.28	Une machine dans tous ses états !	37

7.29	Permutations	37
7.30	Écart type	37
7.31	Décalage d'éléments 1	37
7.32	Décalage d'éléments 2	37
7.33	Interpolation de Lagrange	38
7.34	Modèle SIR discret	38
7.35	Prédire le chaos ?	38
7.36	Chiffrement de Vigenère	39
7.37	« Bonjour-hi »	40
7.38	Ça monte et ça descend !	40
8	Annexe	41
8.1	Code du 7.18	41

1 Syntaxe, variables et I/O

1.1 Explications 1

Pour chaque programme, lister les variables, quel type de données elles peuvent contenir (ex : dire "nombre entier" pour "int") et expliquer brièvement ce qu'il fait.

a.

```
#include <iostream>
using namespace std;

int main()
{
    int age;
    cin >> age;
    cout << "Tu as " << age << " ans.";

    return 0;
}
```

b.

```
#include <iostream>
using namespace std;

int main()
{
    float nombre;
    cin >> nombre;
    cout << "Le carré de " << nombre << " est " << nombre * nombre << " .";

    return 0;
}
```

c.

```
#include <iostream>
using namespace std;

int main()
{
    float dividende, diviseur;
    cin >> dividende >> diviseur;
    cout << "Le quotient de la division de " << dividende << " par " << diviseur << "
        est " << dividende / diviseur << " .";

    return 0;
}
```

1.2 Erreurs 1

Trouver la ou les erreur(s) dans chaque programme. Donner une correction possible.

a.

```
#include <iostream>
using namespace std;

int main()
{
```

```

    cout >> "Hello World !";

    return 0;
}

```

b.

```

#include <iostream>
using namespace std;

int main()
{
    int a;

    cout << "Donner un nombre : \n";
    cin >> a;

    int b;

    cout << a << " multiplié par " << b << " donne " << a * b

    return 0;
}

```

c.

```

#include <iostream>
using namespace std;

int main()
{
    const int a = 0;

    cout << "Donner un nombre : \n";
    cin >> a;

    cout << "Le double de " << a << " est " << 2 * a << " .';

    return 0;
}

```

1.3 Complétions 1

Compléter la ou les lignes manquantes dans chaque programme¹.

a.

```

#include <iostream>
using namespace std;

// On cherche à calculer et afficher le périmètre d'un cercle
int main()
{
    const float pi = 3.14;
    ...

    cout << "Donner le rayon du cercle : \n";
    cin >> rayon;
}

```

1. Chercher au besoin les formules sur Internet.

```

    cout << "Le périmètre du cercle est " << 2 * pi * rayon << ".";

    return 0;
}

```

b.

```

#include <iostream>
using namespace std;

// On cherche à calculer et afficher l'aire d'un triangle
int main()
{
    float base, hauteur;

    cout << "Donner la base et la hauteur du triangle : \n";
    ...

    cout << "L'aire du triangle est : ";
    ...

    return 0;
}

```

1.4 Alice et Bob 1

Bob a écrit le programme suivant :

```

#include <iostream>
using namespace std;

int main()
{
    int a, b;

    cout << "Donner deux nombres : \n";
    cin >> a >> b;

    cout << "Le quotient de " << a << " par " << b << " est " << a / b;

    return 0;
}

```

Lorsqu'Alice teste le programme de Bob avec les valeurs $a = 1$ et $b = 3$ elle n'obtient pas la valeur attendue. Trouver le problème.

1.5 Triple d'un nombre

Écrire un programme qui renvoie le triple d'un nombre.

1.6 Aire d'un cercle

Écrire un programme qui calcule l'aire d'un cercle.

1.7 Une expression... compliquée !

Voici une expression un peu compliquée :

$$\frac{(a+b+c)(a+c)(a+b)(b+c)}{2}$$

Écrire un programme qui calcule cette expression pour n'importe quel a, b et c. Attention aux priorités opératoires !

1.8 Racines carrées

Il est possible de calculer la racine carrée d'une expression en C++. Pour ce faire, il faut en premier temps rajouter la ligne suivante au début du programme :

```
#include <cmath>
```

Supposons que l'on veuille calculer $\sqrt{2}$. La ligne suivante permet de stocker cette valeur dans une variable :

```
float racinecarree = sqrt(2);
```

Écrire un programme calculant la racine carrée de l'expression de l'exercice 1.7.

1.9 Puissances

Cet exercice nécessite le programme du 1.8. Il est possible de calculer la puissance x -ième² d'un nombre en C++. Si l'on veut calculer la puissance 0.3-ième de 2 et garder le résultat dans une variable, on peut écrire :

```
float puissance = pow(2, 0.3);
```

Calculer la puissance 0.5-ième de l'expression de l'exercice 1.7. Que remarquez-vous ?

1.10 Un peu de binaire

Un bit³ est une variable ne pouvant prendre que les valeurs 0 ou 1. Il est possible d'encoder les nombres entiers positifs avec des bits en utilisant les puissances de 2. Pour n bits, les exposants vont de $n-1$ à 0 de gauche à droite. Ainsi, le n -ième bit est relié à la valeur 2^{n-1} , le $n-1$ -ième est relié à la valeur 2^{n-2} ... et le premier bit (tout à droite) est relié à la valeur 2^0 . Un bit valant 0 indique que cette valeur n'est pas prise en compte.

Pour décoder le groupe 1011 on fait :

Bit	1	0	1	1	Total
Puissance de 2	2^3	2^2	2^1	2^0	15
Valeur effective	8	0	2	1	11

Écrire un programme permettant de décoder n'importe quel groupe de 4 bits.

2. Je veux dire "nombre puissance x"

3. De l'anglais *binary digit*

2 Structures conditionnelles et boucles

2.1 Explications 2

Pour chaque programme lister tous les blocs, décrire les conditions d'arrêt des boucles ainsi que les conditions à satisfaire pour rentrer dans les structures conditionnelles. Enfin, expliquer brièvement ce que fait le programme.

a.

```
#include <iostream>
using namespace std;

int main()
{
    int age;

    cout << "Donnez votre âge :\n"
    cin >> age;

    if (age >= 18)
    {
        cout << "Vous êtes majeur";
    }

    else if (age >= 0)
    {
        cout << "Vous êtes mineur";
    }

    else
    {
        cout << "Vous êtes un menteur";
    }

    return 0;
}
```

b.

```
#include <iostream>
using namespace std;

int main()
{
    int age = -1;

    while (age < 0)
    {
        cout << "Donnez votre âge :\n"
        cin >> age;
    }

    if (age % 3 == 0 && (age - 2) % 5 == 1)
    {
        cout << "Votre âge satisfait la condition spéciale";
    }

    return 0;
}
```

```
}
```

c.

```
#include <iostream>
using namespace std;

int main()
{
    int base = 0;
    int a = 1;
    int b = 1;
    int temp = 0;

    for (int i = 0; i < 10; i++)
    {
        temp = b;
        b = a;
        a += temp;

        base += a * i;

        if (base > 100)
        {
            cout << "La suite commence à exploser";
        }
    }

    return 0;
}
```

2.2 Erreurs 2

Trouver la ou les erreur(s) dans chaque programme ainsi que les possibles boucles infinies. Donner une correction possible.

a.

```
#include <iostream>
using namespace std;

// On veut afficher 10 fois "bonjour".
int main()
{
    for (int i = 0; i <= 10; i++)
    {
        cout << "bonjour";
    }

    return 0;
}
```

b.

```
#include <iostream>
using namespace std;

int main()
{
```

```

int nombre;

cout << "Donner un nombre :\n";
cin >> nombre;

while (true)
{
    nombre *= 2;
    cout << nombre << endl;

    if (nombre > 100)
    {
        break;
    }
}

return 0;
}

```

c.

```

#include <iostream>
using namespace std;

int main()
{
    int de;

    cout << "Donner le résultat du dé :\n";

    cin >> de;

    switch (de)
    {
        case 1:
            cout << "C'est 1 !";
        case 2:
            cout << "C'est 2 !";
        case 3:
            cout << "C'est 3 !";
        case 4:
            cout << "C'est 4 !";
        case 5:
            cout << "C'est 5 !";
        case 6:
            cout << "C'est 6 !";
        default:
            break;
    }

    return 0;
}

```

2.3 Complétions 2

Compléter la ou les lignes manquantes dans chaque programme.

a.

```
#include <iostream>
using namespace std;

// On cherche à déterminer si le nombre est premier (divisible seulement par lui-même
// et 1)
int main()
{
    int nombre;
    bool premier = true;

    cin >> nombre;

    for (...)
    {
        if (nombre % i == 0)
        {
            premier = false;
            ...
        }
    }

    if (...)
    {
        cout << "Le nombre est premier";
    }

    ...
    {
        cout << "Le nombre n'est pas premier";
    }

    return 0;
}
```

b.

```
#include <iostream>
using namespace std;

// On cherche à déterminer le nombre entier n tel que  $n^3 = 4n^2$ 
int main()
{
    int n = 1;

    while (...)
    {
        n++;
    }

    cout << "n";

    return 0;
}
```

2.4 Du tableau à la boucle

Écrire une boucle et son contenu en se basant sur le tableau suivant :

# de passage	$a < 0.125$	a
0	false	1
1	false	0.5
2	false	0.25
3	false	0.125
4	true	0.0625

2.5 Parité

Écrire un programme qui détermine si un nombre est pair⁴ ou non.

2.6 Les premiers nombres premiers

Écrire un programme qui affiche tous les nombres premiers entre 2 et 100 (inclus).⁵

2.7 Stan Robotix 1

Lors de la programmation du robot, il sera régulièrement utile de raisonner avec des systèmes qui peuvent avoir plus que deux états distincts. Il est également plus agréable de travailler avec des valeurs plus précises que "1", "2" ou "3". Une structure existe pour nous permettre de créer ces systèmes avec différents états : l'énumération. Supposons que le robot puisse rouler avec trois vitesses définies à l'avance. On peut écrire :

```
enum Vitesse
{
    lente,
    moyenne,
    rapide
};
```

Un nouveau type de variable, le type "Vitesse" est alors créé et utilisable comme n'importe quel type. Plus généralement, on a :

```
enum Système
{
    État1,
    État2,
    ...,
    ÉtatN
};
```

Les types venant d'énumération sont utilisables dans les structures *switch-case*. Écrire un programme qui prend un nombre entier en entrée. Afficher le type de vitesse en utilisant une structure *switch-case*.

- Si le nombre est positif, la vitesse est rapide.
- Si le nombre est négatif, la vitesse est lente.
- Si le nombre est nul, la vitesse est moyenne.

4. Un nombre pair est divisible par 2. Gardez cela en tête !
5. Astuce : utilisez une boucle dans une autre.

2.8 La meilleure des structures conditionnelles

Dans le roman *Le meilleur des mondes*⁶ la société est divisée en cinq castes : *Alpha*, *Beta*, *Gamma*, *Delta*, *Epsilon*. Les *Alpha* sont les plus grands. À l'opposé, les *Epsilon* sont les plus petits. Bien que peu de détails ne soient donnés quant à la correspondance entre taille et caste, on peut raisonnablement la faire ainsi :

- *Alpha* : 1m80 et plus
- *Beta* : 1m75-1m80
- *Gamma* : 1m70-1m75
- *Delta* : 1m65-1m70
- *Elpha* : moins de 1m65

En supposant que la taille soit le seul élément déterminant la caste, écrire un programme qui la détermine en fonction de la taille indiquée en entrée. L'exercice précédent peut être utilisé.

2.9 Nombres aléatoires 1

Il existe plusieurs manières de générer des nombres aléatoires en C++. Il faut d'abord rajouter en début de programme la ligne :

```
#include <random>
```

Dans cet exercice, nous allons nous intéresser à un générateur à "distribution normale". Pour générer un nombre aléatoire de ce type, il faut rajouter les lignes suivantes :

```
random_device systeme{};
mt19937 generateur{systeme()};
normal_distribution<float> distribution{0.0, 1.0};

float nombre = distribution(generateur);
```

Écrire un programme générant 200 nombres de cette manière. Le programme devra séparer les nombres en trois catégories⁷ :

- $x < -1$
- $x > 1$
- $-1 \leq x \leq 1$

Afficher combien il y a de nombres dans chaque catégorie. Y-a t-il une catégorie plus représentée que les autres ?

2.10 Nombres aléatoires 2

Cet exercice se base sur le 2.9. Ajoutons deux nouvelles catégories de cette manière :

- $x < -2$
- $-2 \leq x < -1$
- $-1 \leq x \leq 1$
- $1 < x \leq 2$
- $x > 2$

Compter le nombre d'occurrences dans chaque catégorie. Afficher par des étoiles (*) la distribution dans l'ordre indiqué des catégories.⁸

6. Aldous Huxley, 1932

7. Certains pourraient être tentés d'utiliser les énumérations. Ce n'est pas utile dans ce cas.

8. Astuce : afficher la moitié du nombre d'occurrence pour chaque catégorie.

3 Vecteurs, tableaux et pointeurs

3.1 Tableau ou vecteur ?

Choisir, entre un tableau et un vecteur, le moyen le plus approprié de stocker les données suivantes :

- Tous les nombres premiers entre 1 et 100
- Les 10 premiers nombres premiers entre 1 et 100
- Le mode de vitesse du robot durant les phases de la période autonome
- L'âge des répondants d'un sondage

3.2 À couvert !

On a rangé les précipitations moyennes de pluie et de neige à Montréal entre 1981 et 2010⁹ dans des tableaux. Compléter le programme suivant pour obtenir la moyenne et le total annuel pour chaque tableau.

```
#include <iostream>
using namespace std;

int main()
{
    float pluie[12] = {27.3, 20.9, 29.7, 67.7, 81.2, 87.0, 89.3, 94.1, 83.0, 89.1, 76.7,
                      38.8};
    float neige[12] = {49.5, 41.2, 36.2, 12.9, 0, 0, 0, 0, 0, 1.8, 19.0, 48.9};

    float moyennePluie = 0, moyenneNeige = 0, totalPluie = 0, totalNeige = 0;

    ...
    .
    .
    .
    ...

    return 0;
}
```

3.3 Une racine carrée... compliquée !

Se référer à l'exercice 1.8 pour l'utilisation de la racine carrée. Pour calculer la distance entre deux points A et B de coordonnées respectives (x_a, y_a) et (x_b, y_b) , on utilise la formule suivante :

$$d = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

Par ailleurs, on a rangé les coordonnées de deux points dans des tableaux :

```
float posX[2] = {0.3, -1.2};
float posY[2] = {0.6, 2};
```

Donner la ligne permettant de calculer la distance entre les deux points.

9. Statistiques de la ville de Montréal : <https://ville.montreal.qc.ca/pls/portal/docs/1/85762076.JPG>

3.4 Prédictions 1

Pour chaque programme, déterminer les valeurs de toutes les variables à la fin son exécution.

a.

```
#include <iostream>
using namespace std;

int main()
{
    int a = 2;
    int* b = new int;
    b = &a;
    a = 5;

    return 0;
}
```

b.

```
#include <iostream>
using namespace std;

int main()
{
    int a = 3;
    int* b = new int;
    *b = a;
    *b = 4;

    return 0;
}
```

3.5 Alice et Bob 2

Bob a créé un programme permettant de ranger n'importe quelle quantité de nombres entiers dans un tableau.

```
#include <iostream>
using namespace std;

int main()
{
    int taille;
    cin >> taille;

    int tableau[taille];
    int nombre;

    for (int i = 0; i < taille; i++)
    {
        cin >> nombre;
        tableau[i] = nombre;
    }

    return 0;
}
```

Lorsqu’Alice teste son programme, celui-ci ne fonctionne pas. Trouver le problème et le corriger.

3.6 Un vecteur bien long

Écrire un programme rangeant les $2n + 1$ premiers carrés dans un vecteur, où n est un entier donné par l’utilisateur. Afficher la taille du vecteur et le nombre en son milieu.

3.7 Stan Robotix 2

Se référer à l’exercice 2.7 pour l’utilisation des énumérations. Écrire un programme prenant un entier en entrée et ayant un vecteur de type `Vitesse`. Le programme fait les actions suivantes :

- Le nombre est 1 : on ajoute au vecteur un élément "lente" de type `vitesse`
- Le nombre est 2 : on ajoute au vecteur un élément "moyenne" de type `vitesse`
- Le nombre est 3 : on ajoute au vecteur un élément "rapide" de type `vitesse`
- Le nombre est autre chose : le programme s’arrête.

Si le nombre est 1, 2 ou 3, le programme doit continuer à demander un nombre.

3.8 Céline et les statistiques

Céline a écrit un programme permettant de prédire¹⁰ des mesures à partir d’observations passées. Son programme contient deux vecteurs comme suit :

```
vector<int> occurrences = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
    19, 20};
vector<int> observations = {0.17, 0.26, 0.35, 0.51, 0.55, 0.45, 0.6, 0.63, 0.68, 0.72, 0.79,
    0.86, 0.82, 0.9, 0.88, 0.93, 0.97, 1.0, 1.1, 1.11, 1.15};
```

Céline a également calculé une valeur précise qu’elle stocke dans une variable :

```
float a = 0.4435;
```

Mais Céline a oublié comment calculer la deuxième ! Cette valeur se calcule ainsi :

$$b = \bar{y} - a\bar{x}$$

Où \bar{x} et \bar{y} représentent respectivement la moyenne des occurrences et la moyenne des observations. Écrire un programme calculant la valeur manquante. Prédire la mesure pour 21 occurrences avec la formule suivante :

$$m = ao + b$$

Avec m la mesure et o le nombre d’occurrences.

3.9 Un vecteur bien long... mais sans pairs !

Pour effacer le premier élément d’un vecteur, on peut utiliser la ligne suivante :

```
vecteur.erase(vecteur.begin());
```

Si l’on souhaite effacer l’élément à la position n ¹¹, on peut modifier la ligne en :

```
vecteur.erase(vecteur.begin() + n);
```

10. La méthode utilisée est une régression linéaire

11. Rappel : la position dans un vecteur commence à 0

Ajouter au programme de l'exercice 3.6 un bout de code supprimant tous les nombres pairs du vecteur avant d'afficher sa taille et son premier élément.

4 Fonctions

4.1 Le meilleur des types de retour

Pour chaque proposition, indiquer le type de retour le plus judicieux (plusieurs réponses sont parfois possibles).

- Une fonction déterminant la parité d'un nombre
- Une fonction calculant l'aire d'un cercle en fonction du rayon
- Une fonction déterminant si le robot est à moins d'un mètre du mur
- Une fonction réinitialisant les paramètres du robot
- Une fonction qui affiche un message en fonction de l'heure

4.2 Une fonction... spéciale !

Voici une fonction un peu spéciale :

```
float fonctionSpeciale(float x)
{
    return sqrt((x - 2) / x);
}
```

Cette fonction renvoie des valeurs aberrantes pour $0 \leq x < 2$. Proposer un ajout permettant d'éviter ces valeurs aberrantes.

4.3 Le plus petit des multiples

Écrire et implémenter une fonction à un paramètre (entier) qui détermine le plus petit entier divisible par 17 supérieur au paramètre.

4.4 Mes carrés !

Écrire et implémenter une fonction qui renvoie les 10 premiers carrés sous forme d'un vecteur.

4.5 Céline et le jeu

Céline joue à un jeu qui a les spécificités suivantes :

Gains	0	5	-10
Probabilité	0,2	0,3	0,5

Actuellement, le gain moyen espéré lors d'une partie est de :

$$0 \times 0,2 + 5 \times 0,3 - 10 \times 0,5 = -3,5$$

Céline joue donc à un jeu qui est défavorable. Écrire une fonction à deux paramètres calculant le gain moyen espéré par joueur qui joue à un jeu ayant les spécificités suivantes :

Gains	0	5	g
Probabilité	0,2	0,3	p

Essayer quelques valeurs de g (négatives) et de p (entre 0 et 1) pour voir quelles spécificités seraient favorables.

4.6 In french please !

Un mile vaut environ 1,609 kilomètres. De même, un kilomètre vaut environ 0,621 mile. Écrire une fonction permettant de faire la conversion dans les deux sens pour une mesure donnée.¹²

4.7 Prédiction 2

Pour chaque programme, déterminer les valeurs de chaque variables à la fin de son exécution.

a.

```
#include <iostream>
using namespace std;

void cube(int nombre)
{
    nombre = nombre * nombre * nombre;
}

int main()
{
    int* nombre = new int;
    *nombre = 3;
    cube(*nombre);

    return 0;
}
```

b.

```
#include <iostream>
using namespace std;

void cube(int* nombre)
{
    *nombre = *nombre * *nombre * *nombre;
}

int main()
{
    int nombre = 3;
    cube(&nombre);

    return 0;
}
```

4.8 Nombres aléatoires 3

L'exercice 2.9 présentait un générateur de nombres aléatoires à distribution normale. Les paramètres utilisés étaient 0,0 et 1,0, mais il est possible de les changer. En effet, la ligne suivante est tout à fait valide :

```
normal_distribution<float> distribution{esperance, ecartType};
```

12. Vous pouvez utiliser un deuxième paramètre pour spécifier le sens de conversion.

Il suffit que les variables *esperance* et *ecartType* soit deux *float* bien définis. Écrire et implémenter une fonction permettant de générer des nombres aléatoires à distribution normale mais à paramètres variables.

4.9 SohCahToa !

Il est possible d'utiliser les fonctions trigonométriques en C++. Pour ce faire, il est nécessaire d'inclure le module *cmath* comme pour la racine carrée. Supposons que l'on veuille calculer le cosinus d'un angle, on peut procéder ainsi :

```
float resultat = cos(angle);
```

Notez bien que les fonctions trigonométriques de ce module fonctionnent avec des radians et non pas des degrés classiques.

La fonction suivante fait intervenir la fonction sinus mais n'est pas définie en 0 :

```
float f(float x)
{
    return sin(x) / x;
}
```

Tester la fonction avec de petites valeurs (inférieures à 0,5 mais supérieures à 0) et proposer un ajout permettant d'éviter un résultat aberrant si $x = 0$. Implémenter cet ajout.

4.10 Centaines, dizaines et unités

L'opérateur % permet de séparer un nombre et chaque chiffre qui le compose. En effet, on peut remarquer que :

$$\begin{aligned} 639 \% 10 &= 9 \\ \frac{639 - 9}{10} \% 10 &= 63 \% 10 = 3 \\ \frac{63 - 3}{10} \% 10 &= 6 \% 10 = 6 \end{aligned}$$

Écrire et implémenter une fonction séparant de cette manière un nombre à 3 chiffres. La fonction peut afficher les chiffres séparés par un espace.

4.11 Stan Robotix 3

Il est possible d'obtenir l'heure actuelle en C++. Pour ce faire, il faut rajouter en début de programme la ligne suivante :

```
#include <ctime>
```

Supposons que l'on souhaite obtenir les secondes actuelles. On peut alors écrire :

```
time_t temps = time(0);
tm* heureActuelle = localtime(&temps);
int secondes = heureActuelle->tm_sec;
```

Lors de la période autonome, il peut être utile¹³ de séparer ladite période en sections durant un certain temps. Pour simplifier, on peut résumer les déplacements du robots en 4 catégories :

13. C'est une méthode peu efficace mais simple à mettre en place

- Avancer
- Reculer
- Tourner à droite
- Tourner à gauche

Voici un exemple de période autonome, avec t le nombre de secondes écoulées :

1. $0 \leq t < 10$: avancer
2. $10 \leq t < 12$: tourner à droite
3. $12 \leq t < 20$: avancer
4. $20 \leq t < 28$: reculer
5. $28 \leq t < 34$: tourner à droite
6. $34 \leq t < 40$: avancer
7. $40 \leq t < 46$: reculer
8. $46 \leq t < 48$: tourner à droite
9. $48 \leq t < 58$: reculer
10. $58 \leq t < 60$: ne rien faire

En supposant que la période autonome dure 60s¹⁴, écrire et implémenter une fonction renvoyant l'action que doit faire le robot en fonction des secondes actuelles.

14. Elle dure en réalité 15s.

5 Classes

5.1 Définitions

En utilisant la définition typique des méthodes *get* et *set*, donner une définition possible des méthodes *getX*, *getY*, *setX* et *setY*.

```
class Point
{
    float x, y;

public:
    Point();
    ~Point();

    float getX();
    float getY();

    void setX(float X);
    void setY(float Y);
};
```

5.2 Objet construit !

Cet exercice nécessite la classe du 5.1. On donne une définition possible du constructeur et du destructeur de la classe Point :

```
Point::Point()
{
    cout << "Objet construit !";
}

Point::~~Point()
{
    cout << "Objet détruit !";
}
```

Lister, dans l'ordre, **toutes** les différentes informations affichées par la console lors de l'exécution de la fonction *main* suivante :

```
int main()
{
    Point pt;
    pt.setX(2);
    pt.setY(3);

    cout << 3 * pt.getX() - 4 * pt.getY();

    return 0;
}
```

5.3 Le retour de la racine carrée perdue

Cet exercice nécessite la classe du 5.1. Compléter le programme suivant pour obtenir la distance entre les deux points (voir le 3.3 pour la formule).

```
#include <cmath>
#include <iostream>
using namespace std;

class Point
{
    // Ne pas compléter la classe
};

int main()
{
    Point pt1, pt2;
    pt1.setX(2);
    pt1.setY(3);
    pt2.setX(3);
    pt2.setY(5);

    cout << "La distance entre les points est : " << sqrt(...);

    return 0;
}
```

5.4 Triangle

On utilise un troisième point de coordonnées (4;4) pour former un triangle. Écrire un programme en utilisant la classe *Point* pour déterminer la longueur des trois côtés du triangle, respectivement *a*, *b* et *c*.

5.5 Cercle

Écrire et implémenter une classe cercle ayant les spécificités suivantes :

- Un attribut *rayon*
- Des méthodes *get* et *set* classiques
- Une méthode donnant la circonférence du cercle
- Une méthode donnant l'aire du cercle

5.6 Le fameux hérisson

Sonic est un hérisson bleu courant très vite. Officiellement, il est sensé avoir une vitesse de 342,88 mètres par seconde. La distance parcourue par un objet se déplaçant à vitesse constante pendant une certaine durée est obtenue de la manière suivante :

$$distance = vitesse \times durée$$

Compléter la classe suivante pour déterminer la distance parcourue par Sonic en 2 heures :

```
class Sonic
{
    const float vitesse = ...
}
```

```

public:
    Sonic();
    ~Sonic();

    ...

    ...

    float distanceParcourue(...);
}

...
.
.
.
...

```

5.7 Une architecture classique

En utilisant la classe de l'exercice 5.6, créer deux autres fichiers et répartir le programme en utilisant l'architecture classique :

- sonic.h
- sonic.cpp
- main.cpp

5.8 Jeu 1

On a créé deux fichiers contenant la classe suivante :

```

// joueur.h

enum Action
{
    Cooperer,
    Tricher
};

class Joueur
{
    int score;

public:
    Joueur();
    ~Joueur();

    int getScore();

    void setScore(int);
};

Action choixAleatoire();

```

```

// joueur.cpp

#include "joueur.h"

#include <random>
using namespace std;

Joueur::Joueur() { }
Joueur::~Joueur() { }

int Joueur::getScore()
{
    return score;
}

void Joueur::setScore(int Score)
{
    score = Score;
}

Action choixAleatoire()
{
    int nbr = rand() % 2;

    if (nbr == 0)
    {
        return Action::Cooperer;
    }

    return Action::Tricher;
}

```

L'objectif de cet exercice est de créer une classe modélisant l'affrontement entre deux joueurs. Le jeu possède les caractéristiques suivantes :

Joueur 1		Coopérer	Tricher
Joueur 2			
Coopérer		+3 +3	+5 -1
Tricher		-1 +5	+0 +0

Le joueur 1 joue de manière aléatoire, alors que le joueur 2 commence par coopérer puis copie la dernière action du joueur 1¹⁵. Le nombre de tours durant une partie doit être décidé à l'avance.

- Créer une classe *Jeu* permettant d'encapsuler deux joueurs et le nombre de tours dans une partie.
- Ajouter des méthodes *get* et *set*.
- Créer une méthode *jouer*¹⁶ modélisant l'affrontement entre les joueurs. Afficher à la fin de chaque tour le score de chaque joueur. À la fin de la partie, afficher le joueur gagnant.

15. Astuce : assurez-vous de garder la dernière action dans une variable.

16. La logique de cette fonction pourra faire l'objet d'une discussion au besoin.

5.9 Simulation

Cet exercice nécessite le programme du 5.8. Créer une classe *Simulation* permettant de simuler un nombre indéfini (c'est-à-dire à préciser par l'utilisateur) de parties de 5 tours. Afficher le nombre de victoires de chaque joueur.

5.10 Prédictions 3

Écrire une classe dans un fichier *.h* qui pourrait avoir comme fichier *.cpp* le fichier suivant :

```
#include "sphere.h"
// Le fichier sphere.h ne contient aucune inclusion

Sphere::Sphere() { }
Sphere::~Sphere() { }

float Sphere::getRayon()
{
    return rayon;
}

void Sphere::setRayon(float Rayon)
{
    rayon = Rayon;
}

float Sphere::getAire()
{
    return 4 * pi * rayon * rayon;
}

float Sphere::getVolume()
{
    return 4 / 3 * pi * rayon * rayon * rayon;
}
```

6 Git et Github

6.1 Des commandes et des explications

Associer chaque commande à son utilité :

- a. git pull
 - b. git push
 - c. git fetch
 - d. git commit -m "Ajout d'un fichier"
 - e. git add fichier.h
 - f. git checkout MaBranche
 - g. git status
 - h. git clone
1. Créer un paquet ayant comme titre "Ajout d'un fichier"
 2. Télécharger (sur un serveur) le paquet sur la branche active
 3. Obtenir des informations sur la branche et l'espace de travail local
 4. Basculer sur la branche "MaBranche"
 5. Télécharger (sur son ordinateur) les nouveaux fichiers présents sur la branche
 6. Télécharger (sur son ordinateur) les nouvelles informations de l'espace de travail virtuel
 7. Cloner l'espace de travail virtuel sur son ordinateur
 8. Préparer un paquet en y ajoutant le fichier "fichier.h"

6.2 Entraînement

- a. Créer une branche à partir de *master*.
- b. Ajouter dans le répertoire Activités/2023/ un exercice¹⁷. Bien mentionner son nom.
- c. Télécharger l'exercice sur l'espace de travail virtuel.
- d. Basculer sur la branche d'un autre groupe.
- e. Télécharger l'exercice de l'autre groupe sur l'espace de travail local.
- f. Ajouter un commentaire dans le fichier *main.cpp*.
- g. Télécharger les changements sur l'espace de travail virtuel.
- h. Basculer sur sa branche.
- i. Télécharger les modifications sur l'espace de travail local.
- j. Observer les modications.

17. Décider lequel sur le moment

7 Exercices supplémentaires

7.1 Nombres aléatoires 4

Il est possible de générer des nombres aléatoires et uniformément distribués entre 0 et 1. Pour ce faire, il suffit de changer le générateur (voir l'exercice 2.9) de la manière suivante :

```
uniform_real_distribution<float> distribution{0, 1};
```

Écrire une fonction à un paramètre, que l'on note ici n , et générer n paires de nombres aléatoires entre 0 et 1, notés ici x et y . Après chaque génération de paire, tester la condition suivante :

$$x^2 + y^2 \leq 1$$

Si la condition est remplie, incrémenter un compteur. Sinon, ne rien faire. La fonction devra renvoyer la valeur suivante :

$$4 \times \frac{\text{compteur}}{n}$$

Si la fonction renvoie un nombre entier, forcer la conversion de n en *float*. Que remarquez-vous¹⁸ ?

7.2 Alice et Bob 3

Du haut de ses 553,33 mètres, la tour CN est la plus haute construction humaine au Canada. Bob désire estimer, en négligeant la friction, le temps que prendrait une balle à tomber depuis le haut de la tour. Pour ce faire, Alice prépare la classe suivante :

```
#include <vector>
using namespace std;

class Simulateur
{
    const float g = 9.8067;
    const float hauteurInitiale = 553.33;

    vector<float> positions;

public:
    Simulateur();
    ~Simulateur();

    // Méthodes get et set classiques
    ...
    .
    .
    .
    ...

    void simulation(float);
};
```

La position, en mètres, de la balle en fonction du temps est donnée par l'équation suivante :

$$x(t) = y_0 - \frac{1}{2}gt^2$$

18. La méthode utilisée pour approximer ce nombre s'appelle la méthode Monte-Carlo.

Avec y_0 la hauteur initiale, g l'accélération gravitationnelle et t le temps écoulé, en secondes, depuis le début de la chute.

Bob cherche à déterminer numériquement l'instant où la balle touche le sol, c'est-à-dire où $x(t) = 0$. Alice s'arrête avant d'écrire la définition de *simulation*, mais Bob n'a pas la moindre idée de comment continuer ! Écrire la définition de la méthode *simulation* en tenant compte des choses suivantes :

- Le paramètre de *simulation* représente le nombre de secondes écoulées entre chaque itération. Il est recommandé de tester la méthode avec un paramètre inférieur à 1.
- La méthode devra enregistrer l'ensemble des positions au cours du temps dans le vecteur dédié.
- La méthode devra s'arrêter lorsque la position devient négative et afficher le temps que cela a pris.

7.3 Tri par sélection

Écrire une fonction qui trie (du plus petit au plus grand) un vecteur d'entiers sans utiliser la fonction *sort* de la bibliothèque standard. Conseil pour l'algorithme :

1. Stocker le premier élément du vecteur dans une variable (X).
2. Pour chaque élément du vecteur, déterminer s'il est plus petit que X. Si oui, remplacer X par cet élément.
3. À la fin du parcours du vecteur, ajouter X à un autre vecteur. Supprimer X du premier vecteur (voir l'exercice 3.9 au besoin).
4. Répéter l'algorithme.

7.4 Diviser pour mieux estimer

On a construit la fonction suivante :

```
float f(float x)
{
    return x * x - 2;
}
```

Transcrire l'algorithme¹⁹ suivant en une fonction de paramètre n :

```
a ← 0
b ← 2
Pour i allant de 0 à n :
    Si  $f(a) \times f(\frac{a+b}{2}) < 0$  :
        b ←  $\frac{a+b}{2}$ 
    Sinon :
        a ←  $\frac{a+b}{2}$ 
```

La fonction devra renvoyer $\frac{a+b}{2}$. Que remarquez-vous si vous élevez cette valeur au carré ?

7.5 C'est plus ! C'est moins !

Écrire un programme demandant un nombre entier entre 1 et 100 en entrée. Le programme devra deviner votre nombre le plus efficacement possible en ayant comme seule information l'ordre, c'est-à-dire si son choix est plus petit ou plus grand que le nombre choisi. Afficher le nombre de tentatives requises par l'ordinateur pour trouver votre nombre.

19. Il s'agit d'un algorithme de dichotomie.

7.6 Alors c'est qui le plus malin ?

Faire le même programme que celui du 7.5 en inversant les rôles : vous êtes celui devant deviner un nombre aléatoire entre 1 et 100 en ayant comme seule information l'ordre.

7.7 Stan Robotix 4

Le tableau suivant contient le nombre d'heures de maths par semaine qu'ont 16 personnes reliées (de près ou de loin) à Stan Robotix :

```
float heures[16] = {25, 11, 6, 10, 0, 5, 4, 6, 4, 4, 4, 4, 6, 6, 4, 0}
```

Écrire un programme donnant les valeurs suivantes :

- La moyenne
- La médiane
- Le minimum
- Le maximum

Au besoin, vous pouvez trier un tableau de taille n de la manière suivante :

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    ...
    int tableau[n] = {...};
    ...
    sort(tableau, tableau + n);
    ...
    return 0;
}
```

7.8 Genshin Impact et les chaînes de caractères

Dans le *metagame* du jeu *Genshin Impact*, les différents personnages sont séparés en 3 catégories²⁰.

- *Main DPS* : personnages infligeant la majorité des dégâts dans une équipe et étant longtemps sur le terrain
- *Sup DPS* : personnages infligeant beaucoup de dégâts, mais ayant peu de temps de terrain
- *Support* : personnages ayant diverses facultés, comme le soin, la possibilité de changer des réactions, d'augmenter les statistiques des autres membres de l'équipe...

L'objectif de cet exercice est de renvoyer la catégorie de tous les personnages d'une équipe donnée en entrée.

Il est possible de stocker des chaînes de caractères en C++ de la manière suivante :

```
#include <iostream>
#include <cstring>
using namespace std;
```

```
int main()
```

20. Voir <https://genshin.gg/tier-list>

```

{
    string prenom;

    cout << "Quel est ton prénom ?\n";
    cin >> prenom;
    cout << "Bonjour " << prenom << " !";
}

```

On a rangé les personnages²¹ dans divers vecteurs selon leur catégorie :

```

vector<string> mainDPS = {"Alhaitham", "Ayaka", "Ayato", "Cyno", "Diluc", "Eula", "Ganyu",
    "Hu Tao", "Itto", "Kaveh", "Keqing", "Klee", "Lyney", "Ningguang", "Noelle", "Nomade",
    "Raiden", "Razor", "Tartaglia", "Yanfei", "Yoimiya", "Xiao", "Xinyan"};
vector<string> subDPS = {"Albedo", "Aloy", "Amber", "Beidou", "Fischl", "Heizou", "Kaeya",
    "Tighnari", "Venti", "Xiangling", "Xingqiu", "Yae Miko", "Yelan"};
vector<string> support = {"Baizhu", "Barbara", "Bennett", "Candace", "Chongyun", "Collei",
    "Dehya", "Diona", "Dori", "Faruzan", "Gorou", "Jean", "Kazuha", "Kirara", "Kokomi",
    "Layla", "Lisa", "Lynette", "Mika", "Mona", "Nahida", "Nilou", "Qiqi", "Rosaria",
    "Sara", "Sayu", "Shenhe", "Shinobu", "Sucrose", "Thoma", "Yao Yao", "Yun Jin",
    "Zhongli"};

```

Le programme devra :

1. Demander le nom des personnages composant l'équipe (4 personnages) et les stocker dans un vecteur.
2. Afficher la catégorie de chaque personnage.
3. Afficher (gérer efficacement les différents cas) un message différent en fonction du nombre de *mainDPS*.

7.9 Stan Robotix et les chaînes de caractères

Voir l'exercice précédent pour l'utilisation des chaînes de caractères en C++. On a rangé le nom de certains membres (anciens ou non) de Stan Robotix dans un vecteur. On a fait de même pour leur équipe principale.

```

vector<string> noms = {"Alice", "Alban", "André", "Ines", "Kaloyan", "Majeed", "Max",
    "Maxence", "Maxime", "Mohammad", "Nina", "Raphaël", "Stéphanie", "Victor", "Zachary"};
vector<string> equipes = {"S&D", "Construction", "Programmation", "Construction",
    "Marketing", "S&D", "S&D", "Construction", "S&D", "S&D", "Construction",
    "Programmation", "S&D", "Construction", "Construction"};

```

1. Sélectionner au hasard un nom.
2. Demander la sous-équipe. Si le résultat est bon, augmenter un score.
3. Répéter l'opération au moins 12 fois.

Si l'utilisateur a au moins 70% de bonnes réponses après 12 tentatives, le programme s'arrête. Sinon, le programme continue tant que l'utilisateur n'a pas 70% de bonnes réponses.

21. Versions 4.0 et antérieures. Le voyageur n'est pas compté.

7.10 Neurone artificiel

Les intelligences artificielles comme *ChatGPT* utilisent des modèles d'apprentissage se basant sur un réseau de neurones à plusieurs couches. L'objectif de l'exercice est de modéliser un neurone en C++.

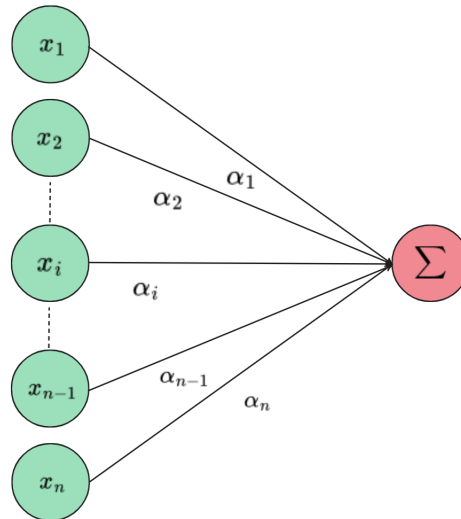


FIGURE 1 – Schéma simplifié d'un neurone artificiel à n entrées

Ce neurone prend n valeurs en entrée et donnera en sortie la valeur suivante :

$$\sum_{i=1}^n \alpha_i x_i = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_i x_i + \dots + \alpha_{n-1} x_{n-1} + \alpha_n x_n$$

On représente souvent les entrées et les poids (les α_i) sous forme de vecteurs. Ainsi, si les entrées sont $X = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$ et les poids $A = \begin{pmatrix} 5 \\ 8 \\ 1 \end{pmatrix}$, le neurone doit renvoyer la valeur :

$$2 \times 5 + 3 \times 8 + 5 \times 1 = 10 + 24 + 5 = 39$$

Écrire et implémenter une classe *Neurone* ayant les spécificités suivantes :

1. Deux vecteurs de *float* représentant les entrées et les poids.
2. Les méthodes *get* et *set*.
3. Une méthode qui renvoie la valeur attendue du neurone.

7.11 En lettres s'il vous plait !

Écrire un programme demandant continuellement un chiffre (entier) à l'utilisateur. Le programme devra l'afficher en lettre.

Exemple : l'utilisateur rentre "2", le programme affiche "deux".

7.12 En chiffres s'il vous plait !

Voir l'exercice 7.8 pour l'utilisation de chaînes de caractères en C++. Chercher comment convertir une chaîne de caractères en nombre *float*, puis écrire un programme demandant en entrée un nombre sous la forme d'une chaîne de caractères. Afficher ensuite la moitié de ce nombre.

7.13 Chiffrement par décalage

Le chiffrement par décalage²² est un exemple très ancien de cryptographie. Il consiste à remplacer chaque lettre d'un message par une autre lettre à distance constante.

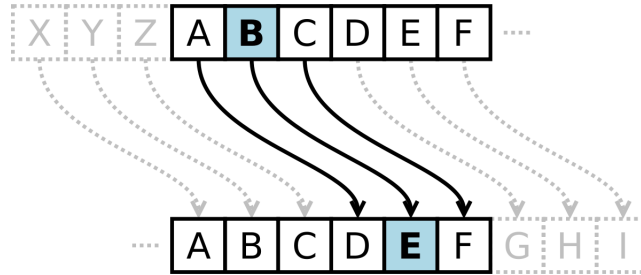


FIGURE 2 – Chiffrement par décalage

Ici²³, le texte est décalé de 3 lettres vers la droite. Écrire un programme ayant les spécificités suivantes :

1. Demander en entrée un nombre entier et une chaînes de caractères (un message en minuscules).
2. Décaler chaque lettre du message en fonction du nombre en entrée. Ne rien faire pour les espaces.
3. Afficher le message chiffré.

7.14 Stan Robotix 5

Le nombre 6622 possède la curieuse particularité d'avoir le même nombre de chiffres que de facteurs premiers. Écrire un programme stockant dans un tableau les facteurs premiers de 6622. Il peut être utile de déterminer tous les nombres premiers entre 2 et 50.

7.15 Brute Force

Voir l'exercice 7.8 pour l'utilisation des chaînes de caractères. Écrire un programme qui génère et affiche toutes les chaînes de caractères possibles de 1 à 4 caractères. Les majuscules, minuscules, chiffres et caractères spéciaux sont à prendre en compte.

7.16 Alice et Bob 4

En finances, l'option *call* constitue un droit d'acheter plus tard un bien à un prix fixé, indépendamment de sa valeur à la date de l'expiration du contrat.

Alice possède une compagnie en pleine effervescence. La valeur d'une action actuellement est de 100\$. Bob désire acheter 100 actions de la compagnie d'Alice, mais dans un an car il n'a pas immédiatement les fonds. Bob va alors signer un contrat *call* avec sa banque, avec un prix d'exercice fixé à 120\$. En échange d'intérêts fixes à verser à la banque, celle-ci assure à Bob de lui fournir dans un an 100 actions de la compagnie d'Alice à 120\$ par action, et ce peu importe le véritable prix de l'action.

On peut donc résumer les différents cas possibles :

22. Voir https://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage

23. Illustration venant de <https://commons.wikimedia.org/wiki/File:Caesar3.svg?uselang=fr>

- En 2024, la valeur d'une action de la compagnie d'Alice dépasse les 120\$: Bob exerce son contrat et Alice est obligé de lui vendre 100 actions à 100\$.
- En 2024, la valeur d'une action de la compagnie d'Alice est en dessous de 120\$: Bob n'a aucun intérêt à exercer son contrat puisqu'il peut acheter les actions sur le marché à un prix inférieur au prix fixé.

L'objectif de cet exercice est de déterminer la somme à verser à la banque à partir de laquelle elle peut espérer faire des gains peu importe la situation.

Valeur supplémentaire	+30\$	−10\$
Probabilité	0.3	0.7

Le tableau ci-dessus présente deux cas de figure, le premier où l'action vaut 130\$ en 2024, l'autre où l'action vaut 90\$ en 2024.

Pour que la banque puisse espérer toujours faire des gains, on démontre que la somme i à verser par action doit satisfaire l'inégalité :

$$i > p\alpha$$

avec p la probabilité d'une augmentation de la valeur de l'action et α la valeur supplémentaire.

Écrire une fonction déterminant numériquement la somme i . La fonction devra pouvoir s'adapter à différentes précisions.

7.17 La pyramide d'astérisques

Voici une tour faite à partir d'astérisques :

```

      *
     ***
    *****
   ********
  
```

FIGURE 3 – Tour de hauteur 4

Écrire un programme qui demande la hauteur à l'utilisateur et qui génère la tour de hauteur associée.

7.18 Un grapheur de base

Le code en annexe permet d'afficher le graphe d'une fonction dans la console. Les abscisses minimales et maximales sont affichées en bas du graphe.

1. Tester le code.
2. Modifier quelques valeurs pour comprendre certaines parties (ne pas toucher au vecteur).
3. Tracer la fonction $f(x) = 2x - 4$.
4. La décaler de 10 unités vers la gauche et mettre le zoom à 1.

7.19 Marche aléatoire

La marche aléatoire est un modèle mathématique de processus reliés au hasard. Le modèle le plus simple consiste à augmenter ou diminuer un compteur de 1 à chaque itération, et ce de façon équiprobable. À l'aide de boucles, simuler la marche aléatoire en partant de 0 pour des durées de 10, 30 et 100 itérations. Faire une moyenne de la position atteinte à la dernière itération.

7.20 Produits de vecteurs

Le produit scalaire de deux vecteurs est la somme du produit terme à terme de chaque élément des vecteurs. Il s'agit d'une opération qui a de multiples applications, notamment en physique pour faire

des calculs sur des forces. Pour $v_1 = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$ et $v_2 = \begin{pmatrix} -2 \\ -2 \\ 4 \end{pmatrix}$, le produit scalaire vaut :

$$\begin{aligned} v_1 \cdot v_2 &= 1 \times (-2) + 3 \times (-2) + 2 \times 4 \\ &= 0 \end{aligned}$$

Dans ce cas (quand le produit scalaire vaut 0), on dit que les vecteurs sont orthogonaux.

Écrire une fonction calculant le produit scalaire de deux vecteurs (attention, ils doivent avoir la même taille).

Il existe un autre produit sur les vecteurs : le produit vectoriel. Celui génère un nouveau vecteur au lieu de renvoyer un nombre. Le vecteur renvoyé est perpendiculaire aux deux autres. L'opération est légèrement plus complexe et se fait uniquement pour des vecteurs à 3 éléments.

Pour $w_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ et $w_2 = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$, le produit vectoriel vaut :

$$w_1 \times w_2 = \begin{pmatrix} bz - cy \\ cx - az \\ ay - bx \end{pmatrix}$$

Écrire une fonction renvoyant le produit vectoriel de deux vecteurs. Vérifier l'orthogonalité du vecteur renvoyé avec les deux premiers.

7.21 Link, un archer hors-pair

Dans la série *The Legend of Zelda*, Link, le protagoniste, est amené à utiliser plusieurs types d'armes, notamment des arcs. Dans cet exercice, on va estimer un angle optimal de visée pour toucher un ennemi situé à une certaine distance. En négligeant les frottements de l'air, on peut montrer qu'un projectile de vitesse initiale v_0 et lancé d'un angle α suit la trajectoire suivante :

$$y(x) = -\frac{9.81}{2} \frac{x^2}{\cos(\alpha)^2 v_0^2} + \tan(\alpha)x$$

où x représente la distance parcourue (en mètres) et $y(x)$ la hauteur du projectile. On peut approximer la vitesse initiale d'une flèche tirée par un arc par 75 m.s^{-1} . Écrire un programme déterminant l'angle optimal pour que Link puisse toucher un ennemi à 420 mètres, avec une précision de 0,1. Link peut-il toucher un ennemi à 600 mètres ? Note : les fonctions \cos et \tan se trouvent dans le module `cmath`.

7.22 Encore plus de binaire

En se basant sur l'exercice 1.10, écrire un programme pour décoder un octet (8 bits). Ensuite, modifier le programme pour décoder n'importe quel nombre de bits.

7.23 Des chiffres mais pas trop

Le concept de chiffres significatifs est intimement lié à celui de mesure. Si vous mesurez la longueur d'un objet à la règle, vous obtiendrez une précision au millimètre. Pour une mesure de 42,0, on dit que le nombre de chiffres significatifs est 3. Utiliser cette valeur dans un calcul et exprimer le résultat avec 6 chiffres n'a alors pas de sens car il serait plus précis que votre mesure. Si l'on veut exprimer

382.93 et 0.008321 avec 3 chiffres significatifs, on obtient 383 (arrondi par excès) et 0.00832 (arrondi par défaut). Notez que les 0 après la virgule ne comptent pas.

Écrire un programme pour exprimer n'importe quel nombre positif avec un nombre spécifié de chiffres significatifs. Indice : pour les nombres plus grand que 1, il peut être utile de le décomposer en unités, dizaines, centaines... (voir l'exercice 4.10). Pour les nombres inférieurs à 1, il peut être utile de savoir la puissance de 10 à partir de laquelle il devient supérieur à 1 (exemple : pour 0,0023 c'est 1000 car $1000 \times 0,0023 = 2,3$).

7.24 Un cercle et des polygones

Le but de cet exercice est d'approximer π en mesurant la longueur de polygones réguliers. Un polygone régulier est un polygone inscrit dans un cercle et dont les côtés sont de la même longueur. En construisant des polygones réguliers dont le nombre de côtés est de plus en plus grand, on peut approximer la circonférence d'un cercle ($2\pi r$) par le périmètre du polygone. Pour un polygone régulier à n côtés inscrit dans le cercle de rayon 1, il est possible de montrer que la longueur d'un côté vaut :

$$l = \sqrt{\left(\cos\left(\frac{2\pi}{n}\right) - 1\right)^2 + \sin^2\left(\frac{2\pi}{n}\right)}$$

Écrire une fonction pour approximer la valeur de π par un polygone régulier à n côtés. Pour imiter le cas pratique où l'on mesure à la règle la longueur d'un côté, limiter la valeur de la racine carrée à 3 chiffres significatifs. Prendre pour valeur de π 3,14159.

7.25 Céline et les diviseurs

Pour un exercice de mathématiques, Céline a besoin de connaître certains diviseurs de 1922310, mais celui-ci en a 96 ! Aider Céline à trouver tous les diviseurs de 1922310 en les stockant progressivement dans un vecteur.

7.26 De l'imagination, il en faut !

Là où les nombres "classiques" peuvent être vus comme des points sur une droite, les nombres complexes peuvent être vus comme des points sur un plan. Les nombres complexes ont aussi la particularité d'assurer la solution de certaines équations polynomiales, notamment $x^2 + 1 = 0$. On note i le nombre tel que $i^2 = -1$. Pour deux nombres complexes $z_1 = a + ib$, $z_2 = x + iy$ avec a, b, x, y des nombres "classiques", les opérations usuelles sont définies comme suit :

$$\begin{aligned} z_1 + z_2 &= (a + x) + i(b + y) \\ z_1 \times z_2 &= (ax - by) + i(bx + ay) \\ \frac{z_1}{z_2} &= \frac{ax + by}{x^2 + y^2} + i \frac{bx - ay}{x^2 + y^2} \end{aligned}$$

Écrire une classe `NombreComplexe` permettant d'encapsuler deux `float`. Écrire ensuite des fonctions `AdditionComplexe`, `MultiplicationComplexe` et `DivisionComplexe` prenant en paramètre deux objets `NombreComplexe` et renvoyant un nombre complexe. Vérifier que $\left(\frac{1+i\sqrt{3}}{2}\right)^2 = -\frac{1+i\sqrt{3}}{2}$ et que $\frac{1+i}{1-i} = i$.

7.27 Jeu 2

Le jeu *Mastermind* consiste à déterminer une série de 4 couleurs différentes (et dans l'ordre) parmi 8 en faisant le moins de tentatives possibles. À chaque essai, le joueur teste une série de couleurs et la machine lui fournit le nombre de couleurs mal placées et le nombre de couleurs bien placées.

Écrire un programme permettant de jouer au *Mastermind*.

7.28 Une machine dans tous ses états !

Se référer à l'exercice 2.9 pour l'utilisation de nombres aléatoires suivant une distribution normale. Une machine à états est un algorithme pouvant fonctionner en boucle fermée (sans intervention humaine) et produisant des résultats différents en fonction d'un état mesuré.

Concevoir une machine à états fonctionnant pendant 100 itérations et générant à chaque fois une mesure aléatoire suivant une loi normale de paramètres 3 et 2.

1. Si la mesure est inférieure à 2, la machine doit "scanner"
2. Si la mesure est entre 2 et 5, la machine doit "actionner"
3. Si la mesure est entre 5 et 9, la machine doit "bouger"

Si la mesure est supérieure à 9, la machine doit s'arrêter prématurément.

7.29 Permutations

Écrire un programme générant toutes les permutations de deux éléments d'un tableau à un nombre fixe éléments.

7.30 Écart type

L'écart type est un nombre mesurant la dispersion des valeurs d'une série statistique par rapport à sa moyenne. L'écart type, noté σ se mesure ainsi :

$$\sigma = \sqrt{\frac{S}{n} - m^2}$$

avec m la moyenne de la série et S la somme des carrés des valeurs de la série. Écrire un programme mesurant la dispersion d'une série statistique stockée dans un vecteur. Quel est l'écart type de la série de l'exercice 7.7 ?

7.31 Décalage d'éléments 1

On peut assimiler un vecteur de 9 éléments à un tableau 3x3, où les 3 premiers éléments sont sur la première ligne, les 3 suivants sur la deuxième, etc... Écrire une fonction permettant de décaler d'un certain nombre de crans les éléments d'une ligne ou d'une colonne d'un tableau 3x3. Par exemple, décaler les éléments de la première ligne de 2 devrait donner :

1	2	3		2	3	1
4	5	6	—————→	4	5	6
7	8	9		7	8	9

7.32 Décalage d'éléments 2

Faire le même programme que l'exercice précédent en utilisant un tableau à deux dimensions au lieu d'un vecteur et une seule fonction au lieu de deux. Indice : il est possible de passer en paramètres un tableau 3x3.

7.33 Interpolation de Lagrange

L'interpolation de Lagrange²⁴ est une manière de faire passer une courbe polynomiale par des points choisis. Pour 3 couples de points (x_1, y_1) , (x_2, y_2) , (x_3, y_3) , l'équation de la courbe est donnée par :

$$f(x) = y_1 \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} + y_3 \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}$$

Écrire une fonction donnant la courbe d'interpolation de 3 couples de points. Déterminer l'image de 4 pour les points suivants : $(0, 1)$, $(3, 1)$, $(-2, 4)$.

7.34 Modèle SIR discret

Le modèle SIR (Sains, Infectés, Rétablis) est un modèle mathématique de propagation d'une maladie dans une population. C'est un modèle à compartiments²⁵ dont on peut facilement lui rajouter ou enlever des catégories. Utiliser le modèle suivant pour écrire une fonction donnant le nombre d'individus sains, infectés et rétablis jour par jour :

1. Chaque individu sain a une probabilité de devenir un nouvel infecté de $\frac{9,437I}{P}\tau$ avec I le nombre d'infectés, P la population totale et τ la probabilité de transmission de la maladie
2. Chaque individu infecté (mais pas nouvel infecté) a une probabilité de guérison γ
3. Les individus rétablis ne peuvent pas être réinfectés
4. À chaque fin de journée, les nouveaux infectés deviennent des infectés "classiques" (ils peuvent à leur tour transmettre la maladie)

Utiliser ce modèle pour prédire l'évolution d'une maladie dans une population de 100000 habitants dont un infecté, avec une probabilité de transmission de 0,25 et une probabilité de guérison de 0,1.

7.35 Prédire le chaos ?

Le mouvement Brownien est un type de mouvement aléatoire décrivant à la base le comportement des particules de l'air. Ce modèle peut aussi être utilisé dans d'autres domaines, notamment en finance où il est utilisé pour prédire les futures tendances boursières. Le programme suivant contient une fonction qui tente de prédire le coût de l'action de Nvidia au cours des 6 prochains mois (seulement les journées de travail) :²⁶

```
#include <iostream>
#include <vector>
#include <cmath>
#include <random>
#include <ctime>
using namespace std;

vector<float> mouvementBrownienGeometrique(float mu, float sigma, float y0, float
    intervalle, float tempsMax)
{
    int iterations = int(tempsMax / intervalle);

    mt19937 generateur;
    generateur.seed(time(0));
    normal_distribution<float> gauss(0, sqrt(intervalle));
```

24. Voir https://fr.wikipedia.org/wiki/Interpolation_lagrangienne

25. Voir https://fr.wikipedia.org/wiki/Mod%C3%A8les_compartimentaux_en_%C3%A9pid%C3%A9miologie

26. Le prix de base de l'action se base sur son prix atteint le 30 avril 2024 à l'ouverture. La volatilité historique a été déterminée sur la période du 23 avril 2023 au 23 avril 2024.

```

vector<float> valeurs;
valeurs.push_back(std::exp((mu - sigma * sigma / 2) * intervalle + sigma *
    gauss(generateur)));

for (int i = 0; i < iterations - 1; i++)
{
    valeurs.push_back(valeurs.at(i) * exp((mu - sigma * sigma / 2) * intervalle + sigma *
        gauss(generateur)));
}

for (int i = 0; i < iterations; i++)
{
    valeurs.at(i) *= y0;
}

return valeurs;
}

int main()
{
    vector<float> valeurs = mouvementBrownienGeometrique(0.12, 0.41, 871.2, 0.000403, 0.05);

    for (int i = 0; i < valeurs.size(); i++)
    {
        cout << valeurs.at(i) << endl;
    }

    return 0;
}

```

En utilisant ce programme, faire une moyenne du prix de l'action de Nvidia après 6 mois sur 20, 100 et 1000 simulations.

7.36 Chiffrement de Vigenère

Le chiffrement de Vigenère est une méthode de chiffrement un peu plus robuste que celui par décalage. Dans ce cas, les lettres du messages sont décalées d'un nombre différents à chaque fois. Ce nombre est déterminé par un mot, appelé clé du message. Par exemple, pour chiffrer le message "persona est une serie de jeux videos japonais" avec la clé "robotique", on procède de la manière suivante :

Persona est une serie de jeux videos japonais
 robotiq uer obo tique ro boti querob otiquero

Le nombre de décalage est déterminé par la position dans l'alphabet de la lettre de la clé associée. Le message chiffré devient alors :

persona est une serie de jeux videos japonais
 robotiq uer obo tique ro boti querob otiquero
 gssghvq ywk ios lmhci us ksnf lchvct txxehezg

Écrire un programme chiffrant un message par selon cette méthode. Indice : il peut être utile de chercher une méthode pour récupérer la taille d'une chaîne de caractères. Également, utiliser la méthode *getline* est une meilleure idée pour récupérer une entrée contenant des espaces.

7.37 « Bonjour-hi »

Écrire un programme agissant comme un caissier : en fonction du prix d'achat d'un produit et de l'argent remis par le client, il doit remettre de la monnaie avec le moins de coupures possibles. Les coupures possibles sont 100\$, 50\$, 20\$, 10\$, 5\$, 2\$, 1\$, 25 cents, 10 cents et 5 cents. On suppose que tous les prix d'achat sont une combinaison de ces coupures.

7.38 Ça monte et ça descend !

Un circuit LC série est un circuit composé d'un condensateur et d'une bobine branchés en série. Ce type de circuit a de nombreuses applications, notamment dans la réalisation de filtres. La tension du courant dans ce genre de circuit oscille entre deux valeurs périodiquement et ce, selon une équation différentielle (l'inconnue est une fonction). Par des techniques d'analyse numérique, on peut montrer qu'une bonne approximation de la tension est donnée par :

$$u_{n+2} = 2u_{n+1} - u_n + h^2 \frac{E - u_n}{LC}$$

avec u_{n+2} , u_{n+1} et u_n les approximations de la tension aux temps (en secondes) $(n+2)h$, $(n+1)h$ et nh , h le pas entre chaque approximation, E la tension aux bornes d'un régulateur de voltage, C la capacité du condensateur et L l'inductance de la bobine.

Écrire un programme stockant dans un vecteur les approximations de la tension jusqu'au temps $t = 10s$ avec un pas de 0,0005 et $u_0 = 0$, $u_1 = 0,00005$. Les valeurs du circuit sont les suivantes : $E = 12V$, $L = 0,5H$, $C = 0,05F$. Sachant que la tension est sensée osciller entre 0 et 24V, l'approximation est-elle bonne (déterminer le maximum / minimum) ? Essayer différente valeur de h pour trouver une meilleure approximation.

8 Annexe

8.1 Code du 7.18

```
#include <cmath>
#include <iostream>
#include <cstring>
#include <vector>

using namespace std;

void affiche(vector<string> grille)
{
    for (int i = 0; i < grille.size(); i++)
    {
        std::cout << grille.at(i) << std::endl;
    }
}

float f(float x)
{
    return cos(x * x) - sin(x);
}

int main()
{
    vector<string> grille;

    for (int i = 0; i < 41; i++)
    {
        grille.push_back(" | ");
    }

    grille.push_back(" ");

    int x = 41, y = 20, zoom = 2;
    float temps = -10, precision = 0.04 / (zoom * zoom);
    float minX;

    int posX = 0, posY = 0;

    bool premiereValeur = true;

    while(true)
    {
        posX = (int)round(x + zoom * 7 * temps);
        posY = (int)round(y - zoom * 5 * f(temps));

        if (posX > 81)
        {
            break;
        }

        if (posY > 0 && posY < grille.size() - 1 && posX > 0)
```

```

    {
        if (premiereValeur)
        {
            minX = temps;

            premiereValeur = false;
        }

        grille.at(posY)[posX] = ' .';
    }

    temps += precision;
}

grille.at(grille.size() - 1).replace(0, to_string(minX).length() - 1, to_string(minX));
grille.at(grille.size() - 1).insert(grille.at(0).length() - to_string(temps).length() -
    1, to_string(temps));

affiche(grille);

return 0;
}

```
