Chapitre 1 : Introduction

Exercice 1.1 Erreurs de syntaxe

Le programme ci-dessous contient plusieurs erreurs de syntaxe.

Proposez un correctif en regard de chaque ligne fautive.

```
/* programme avec erreurs
include iostream;
use spacename std;
int Main()
   out < 'Hello' < endln;
   Return;
end;
```

Correctif

.....
.....
.....
.....
.....
.....
.....

Solution exercice 1.1

```
/* programme avec erreurs

include iostream;
use spacename std;
int Main()
   out < 'Hello' < endln;
   Return;
end;
```

Correctif

```
// programme sans erreurs
/* programme sans erreurs */
#include <iostream>
using namespace std;
int main() {
   cout << "Hello" << endl; // ou cout << "Hello\n";
   return 0; // ou mieux : return EXIT_SUCCESS;
}
```

Exercice 1.2 Un petit dessin

Ecrire un programme C++ permettant de reproduire à l'identique le dessin suivant :

```
  /////
 +-----+
(| o o |)
 |  ^  |
 | '-' |
 +-----+
```

Solution exercice 1.2

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

int main() {

   cout << "  /////"   << endl;
   cout << " +-----+"  << endl;
   cout << "(| o o |)" << endl;
   cout << " |  ^  |"  << endl;
   cout << " | '-' |"  << endl;
   cout << " +-----+"  << endl;

   return EXIT_SUCCESS;
}
```

ou

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

int main() {

   cout << "  /////"   << endl
        << " +-----+"  << endl
        << "(| o o |)" << endl
        << " |  ^  |"  << endl
        << " | '-' |"  << endl
        << " +-----+"  << endl;

   return EXIT_SUCCESS;
}
```

Exercice 1.3 Composition atmosphérique des planètes

Notre système solaire est composé de 8 planètes : Mercure, Vénus, Terre, Mars, Jupiter, Saturne, Uranus et Neptune (dans l'ordre de leur distance au Soleil).

Les 4 premières planètes (Mercure, Vénus, Terre, Mars) sont dites "telluriques".

Les 4 dernières planètes (Jupiter, Saturne, Uranus et Neptune) sont dites "gazeuses".

Toutes les planètes, hormis Mercure, ont une atmosphère.

L'atmosphère de Vénus et de Mars est principalement constituée de CO_2 (dioxyde de carbone) et en moindre quantité de N_2 (diazote). Celle de la Terre est, elle, principalement constituée de N_2 et en moindre quantité de O_2 (dioxygène). Enfin, l'atmosphère des planètes gazeuses est, elle, principalement constituée de H_2 (dihydrogène) et en moindre quantité de He (hélium).

Ecrire un programme C++ qui affiche l'ensemble des informations données ci-dessus sous forme d'un tableau à 4 colonnes (type de la planète, nom de la planète, gaz principal et gaz secondaire).

Solution exercice 1.3

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

int main() {

   cout << "+-----------------------------------------------------------+" << endl
        << "| Type planete | Nom planete | Gaz principal | Gaz secondaire |" << endl
        << "|-----------------------------------------------------------|" << endl
        << "| Tellurique   | Mercure     | -             | -              |" << endl
        << "| Tellurique   | Venus       | CO2           | N2             |" << endl
        << "| Tellurique   | Terre       | N2            | O2             |" << endl
        << "| Tellurique   | Mars        | CO2           | N2             |" << endl
        << "| Gazeuse      | Jupiter     | H2            | He             |" << endl
        << "| Gazeuse      | Saturne     | H2            | He             |" << endl
        << "| Gazeuse      | Uranus      | H2            | He             |" << endl
        << "| Gazeuse      | Neptune     | H2            | He             |" << endl
        << "+-----------------------------------------------------------+" << endl;

   return EXIT_SUCCESS;
}
```

Exercice 1.4 Votre année de naissance

Ecrire un programme C++ qui :

1. Demande à l'utilisateur d'entrer son prénom
2. Lit la réponse de l'utilisateur (on supposera celle-ci correcte) et la stocke dans une variable `prenom` de type *string*
3. Demande à l'utilisateur d'entrer son âge
4. Lit la réponse de l'utilisateur (on supposera celle-ci correcte) et la stocke dans une variable `age` de type entier *int*
5. Calcule l'année de naissance (à un an près) de l'utilisateur et l'enregistre dans une variable `annee_naissance` de type entier *int*
6. Affiche à l'écran le message suivant :
   `Bonjour <prenom>,`
   `Vous avez <age> ans et vous êtes ne(e) en <annee_naissance>.`

Indications

- Pour lire le prénom :
  - ajouter la ligne `#include <string>` avant le *main*
  - insérer les 2 lignes de code suivantes dans le *main*
    ```
    string prenom;
    getline(cin, prenom);
    ```
- Pour lire l'âge, utiliser l'instruction : `cin >> age;`

Solution exercice 1.4

```
#include <iostream>
#include <string>

using namespace std;

int main() {

   string prenom;
   cout << "Quel est votre prenom ? ";
   getline(cin, prenom);

   int age;
   cout << "Quel age avez-vous ? ";
   cin >> age;

   const int ANNEE_COURANTE = xxx; // xxx à remplacer par l'année courante

   cout << "Bonjour " << prenom << "," << endl
        << "Vous avez " << age << " ans et vous etes ne(e) en "
        << ANNEE_COURANTE - age << "." << endl;

   return EXIT_SUCCESS;
}
```

Forme plus sophistiquée (mais hors de portée des étudiants pour l'instant) :

```
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <string>

using namespace std;

int main() {

   string prenom;
   cout << "Quel est votre prenom ? ";
   getline(cin, prenom);

   int age;
   cout << "Quel age avez-vous ? ";
   cin >> age;

   // L'année courante est récupérée à partir de la date-heure du système
   char buffer[5];
   time_t t = time(NULL);
   strftime(buffer, sizeof(buffer), "%Y", localtime(&t));
   const int ANNEE_COURANTE = stoi(string(buffer));

   cout << "Bonjour " << prenom << "," << endl
        << "Vous avez " << age << " ans et vous etes ne(e) en "
        << ANNEE_COURANTE - age << "." << endl;

   return EXIT_SUCCESS;
}
```

Exercice 1.5 Train ou voiture ?

Ecrire (en pseudo-code) l'algorithme permettant de traiter le problème suivant :

Vous souhaitez déterminer s'il est plus intéressant (du point de vue coût) de vous rendre de chez vous à votre lieu de travail en voiture ou en train. Les informations connues sont :

- *la distance en km séparant votre domicile de votre lieu de travail*
- *le coût du billet de train simple course*
- *la consommation (litres aux100 km) de la voiture*
- *le prix du litre d'essence*
- *les coûts d'amortissement (Frs par km) de la voiture*

Solution exercice 1.5

Entrées

```
distance_domicile_travail (km)
prix_billet_train (Frs)
consommation_essence (litres aux 100 km)
coûts_amortissement_voiture (Frs par km)
prix_litre_essence (Frs)
```

Sortie

```
Déterminer quel est le moyen de transport (train ou voiture) qui coûte le moins cher pour
se rendre de son domicile à son lieu de travail
```

Algorithme

```
coût_trajet_voiture =
   distance_domicile_travail * (consommation_essence * prix_litre_essence / 100
                                + coûts_amortissement_voiture)
Si coût_trajet_voiture < prix_billet_train
   Choix = voiture
Sinon si coût_trajet_voiture = prix_billet_train
   Choix = train ou voiture
Sinon
   Choix = train
```

Vérification

```
distance_domicile_travail = 100
prix_billet_train = 70
consommation_essence = 5.1
coûts_amortissement_voiture = 0.55
prix_litre_essence = 1.48
coût_trajet_voiture = 100 * (5.1 * 1.48 / 100 + 0.55) = 62.548 < prix_billet_train
=> choix = voiture
```

Exercice 1.6 Compte bancaire (1)

Ecrire (en pseudo-code) l'algorithme permettant de traiter le problème suivant :

Initialement un compte bancaire possède un solde de 10'000.-. Le taux d'intérêt annuel de ce compte est de 6%. A la fin de chaque mois, les intérêts sont capitalisés et un retrait de 500.- est effectué. Au bout de combien de mois, le compte sera-t-il à découvert ?

Solution exercice 1.6

Entrées

```
solde_initial = 10000
taux_intérêt_annuel = 0.06
retrait_mensuel = 500
```

Sortie

```
Afficher au bout de combien de mois le compte est à découvert
```

Algorithme

```
solde_compte = solde_initial
taux_intérêt_mensuel = (1 + taux_intérêt_annuel)^(1/12) – 1   (*)
nb_mois = 0
Tant que solde_compte >= 0
   nb_mois = nb_mois + 1
   solde_compte = solde_compte * (1 + taux_intérêt_mensuel) – retrait_mensuel
Afficher "Le compte sera à découvert au bout de nb_mois"
```

Vérification

…

```
(*)
La formule garantit que :
solde_initial * (1 + taux_intérêt_annuel)
= solde_initial * pow(1 + taux_intérêt_mensuel, 12);

ATTENTION !
Serait faux de définir :
taux_intérêt_mensuel = taux_intérêt_annuel / 12
```

Exercice 1.7 Compte bancaire (2)

Reprenons l'exercice 1.6 et supposons maintenant que le montant initial du compte, le taux d'intérêt annuel ainsi que le montant du retrait mensuel soient des données, non plus fixes, mais fournies par l'utilisateur.

Réécrire l'algorithme de l'exercice 1.6 en tenant compte de cette nouvelle donne.
NB On supposera que chacune des 3 valeurs fournies par l'utilisateur est ≥ 0.

Solution exercice 1.7

```
Entrées
solde_initial = valeur fournie par l'utilisateur
taux_intérêt_annuel = valeur fournie par l'utilisateur
retrait_mensuel = valeur fournie par l'utilisateur

Sortie
Afficher au bout de combien de mois le compte est à découvert

Algorithme
solde_compte = solde_initial
taux_intérêt_mensuel = (1 + taux_intérêt_annuel)^(1/12) - 1
nb_mois = 0
Si solde_compte * taux_intérêt_mensuel >= retrait_mensuel
   Afficher "Le compte ne sera jamais à découvert"
Sinon
   Tant que solde_compte >= 0
      nb_mois = nb_mois + 1
      solde_compte = solde_compte * (1 + taux_intérêt_mensuel) – retrait_mensuel
   Afficher "Le compte sera à découvert au bout de nb_mois"

Vérification
…
```