L'erreur de syntaxe [pr01] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 31 mai 2018

Table des matières

C++ - L'erreur de syntaxe (Solution)

Mots-Clés Programmation ■
Durée estimée 10 min ■

Objectif
Cet exercice consiste à corriger des erreurs de syntaxe.

...(énoncé page suivante)...

1 Énoncé

Types d'erreurs

Elles sont catégorisées en :

- **Erreurs de syntaxe** : le programme ne respecte pas les règles syntaxiques du langage. Ces erreurs sont relativement faciles à trouver car le compilateur signale toujours le problème en indiquant souvent l'endroit de l'erreur, parfois quelques lignes après.
- **Erreurs d'implémentation** (ou *erreurs d'exécution*) : la syntaxe du programme est correcte (le compilateur le compile sans erreur) mais ce que fait le programme est erroné (c.-à-d. une division par zéro se produit, une variable n'a pas été initialisée correctement,..). Ce type d'erreurs ne se détecte que lors des tests, soit par un arrêt intempestif du programme (cas de la division par zéro), soit par la production de résultats erronés (cas de la mauvaise initialisation). Ces erreurs sont plus difficiles à trouver car elles ne sont pas toujours visibles !
- **Erreurs d'algorithmes** : l'algorithme ne se comporte pas comme on s'y attendait. Ce type d'erreur est assez proche du précédent sauf qu'ici, c'est la méthode qui est erronée alors que dans le cas précédent il s'agit d'une étourderie ou d'un manque de précision dans une des étapes de l'algorithme. Des tests formels peuvent être effectués (en tous cas en théorie) pour trouver ce type d'erreurs. Mais dans la pratique, les erreurs algorithmiques sont plutôt découvertes au moyen de tests systématiques... et de cellules grises !
- **Erreurs de conception** : l'algorithme (la « recette ») ne fait pas ce que l'on croit (ce qu'il devrait). C'est alors la méthode globale, voire même l'approche du problème qui est erronée souvent en raison d'hypothèses trop fortes ou non explicitées. Ce niveau relève de l'ingénierie informatique et du « génie logiciel ».

Objectif

On ne s'intéresse ici qu'aux erreurs de syntaxe. De telles erreurs sont fréquemment commises : c'est normal ! Il faut donc apprendre à connaître leurs origines et comprendre le sens des messages du compilateur qui peuvent être très utiles pour corriger rapidement un programme.

2 Programmation

Fichier projet

Créez un projet (nommé `qzxbonjour1` par exemple), puis pour chacun des programmes ci-dessous :

1. Copiez/collez le programme depuis le fichier PDF dans votre environnement.
2. Compilez le programme : un (ou des) message(s) d'erreur(s) indiquera(ront) la ligne où s'est produite l'erreur : corrigez et recompilez.

Indices et commentaires

- Pour aller à la ligne numéro nnn, double-cliquez sur la ligne du message d'erreur.
- Quand on débute les messages d'erreurs sont souvent « ésotériques ». Avec un peu d'expérience, ils le deviennent rapidement beaucoup moins, en tout cas la plupart !
- Certaines situations peuvent dérouter le compilateur au point qu'il affiche plusieurs messages d'erreurs pour une seule erreur. **Conseil** Corrigez toujours les erreurs dans l'ordre dans lequel elles sont données par le compilateur.
- Les couleurs des différentes parties du programme peuvent vous aider à trouver les erreurs.

(1a)

Une erreur de syntaxe :

```
#include <iostream>
using namespace std;
int main()
  cout<<"Bonjour tout le monde !"<<endl;
}
```

Solution C++

Oubli de l'accolade ouvrante de la fonction `main`.

(1b)

Une erreur de syntaxe :

```
#include <iostream>
using namespace std
int main()
{
  cout << "Bonjour tout le monde !" << endl;
}
```

Solution C++

Oubli du point-virgule après l'instruction `using namespace std;`

(1c)

Une erreur de syntaxe :

```
#include <iostream>
using namespace std;
int main(int, Char*[])
{
  cout << "Bonjour tout le monde !" << endl;
}
```

Solution C++

Le type `char` doit commencer par une minuscule.

(1d)
Une erreur de syntaxe :

```
#include <iostream>
using namespace std;
int main()
{
  cout << "Bonjour tout le monde ! << endl;
}
```

Solution C++
Constante littéral chaîne mal fermée.

(1e)
Une erreur de syntaxe :

```
#include <iostream>
using namespace Std;
int main()
{
  cout << "Bonjour tout le monde !" << endl;
}
```

Solution C++
Le symbole `std` doit commencer par une minuscule.

(1f)
Une erreur de syntaxe :

```
#include <iostream>
using namespace std;
int main()
  cout << "Bonjour tout le monde !" << endl;
```

Solution C++
La fonction `main` doit avoir un bloc (des accolades `{` et `}`) autour de ses instructions.

(1g)
Une erreur de syntaxe :

```
#include <iostream>
using namespace std;
int Main()
{
  cout << "Bonjour tout le monde !" << endl;
}
```

Solution C++
L'identifiant `main` doit commencer par une minuscule.

(1h)
Une erreur de syntaxe :

```
#include <iostream>
using namespace std;
{
  cout << "Bonjour tout le monde !" << endl;
}
```

Solution C++
La ligne qui indique l'en-tête de la fonction `main` ne doit pas être en commentaire.

(1i)
Une erreur de syntaxe :

```
#include <iostream>
using namespace std;
int main()
{
  cout << '"';
  cout << " Bonjour tout le monde ! ";
  cout << """ << endl;
}
```

Solution C++
Guillemet mal fermé dans le dernier `cout`.

(1j)
Deux erreurs de syntaxe :

```
#include <iostream>
using namespace std;
int main()
{
  cout << "\"Bonjour\n
          "tout\nle\nmonde !\"" << endl
}
```

Solution C++
Guillemets mal fermés dans l'instruction `cout` (1re ligne) et manque le point-virgule.

(1k)
Deux erreurs de syntaxe :

```cpp
#include <iostream>
using namespace std;
int main()
{
  cout >> "Bonjour tout le monde !" << end;
}
```

Solution C++

Chevrons dans le mauvais sens `cout<<` et le symbole du retour de ligne est `endl`.

(1l)

Dans ce dernier programme, il y a plusieurs erreurs de syntaxe.
Corrigez et recompilez jusqu'à ce qu'il soit exempt d'erreur.

```cpp
include <iostream>
INT main
{
  cout >> "Bonjour tout le monde !"endl
```

Solution C++

Dans l'ordre de correction des erreurs citées par le compilateur :

- Manque `#` devant `include`.
- Le mot-clé `int` doit être écrit en minuscules.
- Manque `using namespace std;` (ce qui permet de reconnaître `cout`).
- Manque le point-virgule à la fin de l'instruction.
- Chevrons dans le mauvais sens `cout<<`.
- Manque les chevrons entre le guillemets fermant et `endl`.
- Manque les parenthèses de `main()`.
- Manque l'accolade fermante de `main()`.

3 Références générales

Comprend [] ■