Introduction à la programmation

M1 2024-2025

Travaux Pratiques 1

Erreur et avertissement à la compilation

Il existe principalement cinq types d'erreurs :

- **Syntax error :** Ces erreurs sont principalement dues à des fautes de frappe ou au non-respect de la syntaxe du langage de programmation spécifié.
  Une telle erreur se produit si le point-virgule (;) est absent à la fin de la déclaration.
- **Run-time error :** Lorsque le programme est en cours d'exécution, et qu'il n'est pas en mesure d'effectuer l'opération.
  La division par zéro est un exemple courant de cette erreur.
- **Linker error :**Ces erreurs sont principalement générées lorsque le fichier exécutable du programme n'est pas créé. Cela peut être dû à un mauvais prototypage de la fonction ou à l'utilisation d'un mauvais fichier .h.
  Une telle erreur consiste à utiliser Main() au lieu de main().
- **Logical error :** L'erreur logique est une erreur qui conduit à une sortie non souhaitée.
- **Semantic error :** Les erreurs sémantiques sont les erreurs qui se produisent lorsque les déclarations ne sont pas compréhensibles par le compilateur.
  L'utilisation d'une variable non initialisée produit une telle erreur.

Exercice 1. *Erreur*

Essayez de répondre aux questions suivantes pour tous les codes de cet exercice :

1. Que fait la fonction suivante ?
2. Quel type d'erreur présente-t-elle lorsque vous la compilez ?
3. Que faut-il faire pour ne pas avoir de message d'erreur ?

Code A :

```c
#include <stdio.h>
int main(void) {
    int a=2;
    int b=2/0;
    int c = a + b;
    printf("The value of c is : %d", c);
    return 0;
}
```

Code B :

```c
#include <stdio.h>
int Main(void) {
    int a=100;
    printf("The value of a is : %d", a);
    return 0;
}
```

Code C :

```c
#include <stdio.h>
int main(void){
    int a = 2;
    int b = 3;
    int c = 1;
    a * b = c;
    return 0;
}
```

Code D :

```c
#include <stdio.h>
int main(void) {
    int a = 200;
    printf("La valeur de a est : %d", a):
    return 0;
}
```

Variables

Dans tout programme, le programmeur fait appel à des données qui prendre des valeurs et qui peuvent changer pendant toute l'exécution du programme. On parle alors de *variables*. Une variable a un type qui détermine les valeurs que la variable peut prendre ainsi que les opérations pouvant s'y appliquer. Nous utiliserons les types standards de C suivant : *int*, *float* et *char*.

Les fonctions *printf()* et *scanf()*

La fonction ***printf()*** est utilisée pour transférer du texte, des valeurs de variables ou des résultats d'expressions vers le fichier de sortie standard stdout (par défaut l'écran). Nous utilisons la fonction printf() avec le spécificateur de format %d pour afficher une variable entière. Nous utilisons %c pour afficher un caractère, %f pour une variable flottante, %s pour une variable de type chaîne, %lf pour un double et %x pour une variable hexadécimale.

La fonction ***scanf()*** est l'une des fonctions les plus utilisées pour prendre les données de l'utilisateur. La fonction *scanf()* lit l'entrée formatée à partir de l'entrée standard telle que les claviers.
L'application la plus simple de *scanf()* ressemble à ceci :

```c
#include <stdio.h>
int main(void) {
    int number;
    printf("Entrez un nombre entier : \n");
    scanf("%d", &number);
    printf("Votre nombre entier = %d",number);
    return 0;
}
```

ATTENTION :
Vous devez mettre **&** devant la variable utilisée dans scanf. La raison en sera claire lorsque vous aurez appris à connaître les pointeurs.

Exercice 2. *Variables*

Quelle est la sortie du programme C donné ? Pouvez-vous l'expliquer ?

```c
#include<stdio.h>
int main(void) {
  int a = 5;
  int b = 2;
  printf("%d", a / b);
  return 0;
}
```

Exercice 3. *Variables*

Expliquez le role de chaque ligne du programme suivant. Quel type d'erreur présent-t-elle lorsque vous la compiler ? Que faut-il faire pour ne pas avoir de message d'erreur ? Avez-vous besoin de toutes les lignes ?

```c
#include <stdio.h>

int main(void) {
    int a = 5;
    int b = 2;
    int c = (a / b);
    printf("Entrez deux nombres entiers : \n");
    // Taking multiple inputs

    scanf("%d %d", &d, &e);
    int s = a + d + e;
    int m = s - a;
    printf("%d", m);
    return 0;
}
```

Exercice 4. *Création de votre premier programme*

Créer un répertoire TP1 et vous y placer. A l'aide de l'éditeur de votre choix, écrire dans un fichier dont le nom est NOM.c, un programme C qui affiche votre nom, votre prénom et votre numéro numéro d'étudiant. Compiler ce programme.

1. Quel est le résultat de la compilation si aucun paramètre autre que le nom du fichier à compiler n'est fourni ?
2. Comment vérifier que l'exécution est correcte ?
3. Comment donner le nom *firstProgram* au fichier exécutable ?

Exercice 5. *Jours vers années, mois, semaine, jours*

Placer vous dans le répertoire TP1. Ecrire un programme *jours.c* transformant un nombre de jours J entré par l'utilisateur en nombre d'années, de mois, de semaines et de jours restant.
Pour simplifier le problème, on considèrera que tous les mois ont 30 jours et toutes les années 360 jours.
L'affichage se fera sous la forme : *J jours correspondent a : xx annee xx mois xx semaine xx jours !*

Exemple :
»Entrez un nombre de jours : 738
»738 jours correspondent a : 2 annees 2 semaines et 4 jours !

Exercice 6. *Fahrenheit vers Celsius*

Pour convertit des degrés Fahrenheit en degrés Celsius, on a la formule suivante :

$$C = 0.55556 \times (F - 32)$$

où F est une température en degrés Fahrenheit et C la temperature équivalente en dégres Celsius.

1. Ecrire un programme *FahrenheitCelsius.c* qui convertit une température en Fharheneit, entrée par l'utilisateur, vers une température en Celsius.
   Indice : Pour mémoriser dans une variable tempF un nombre réel vous devrez utiliser le type ***float***

   Exemple :
   » Entrez température en Fahreneit : 55
   » 55 F correspond à : 12.77788 C
2. Ecrire un programme pour la conversion inverse.