C++ TP 8 exceptions, valeurs par défaut, surcharge

Partie 1 Surcharge et valeurs par défaut

Exercice 1 (sur papier)

Soient les déclarations suivantes :

```
a. void f(int) ;
b. void f(double) ;
c. void f(int, double) ;
d. void f(double, int) ;
e. char   c;
f. int    i;
h. double d;
```

Les appels suivant sont-ils corrects ? Si oui quelles sont les fonctions appelées et les conversions implicites effectivement réalisées ?

```
1.      f(i);
2.      f(d);
3.      f(c);
4.      f(i,d);
5.      f(d,i);
6.      f(i,i);
7.      f(i,c) ;
```

Exercice 2 (sur papier)

Quels sont affichages produits par ce programme ? Pourquoi ?

```
#include <iostream>
using namespace std;
int f(int);
double f(int);
int main(){
 int i;
 double d;
 i = f(2);
 d = f(4);
 cout <<" i = "<< i <<" et d = "<< d <<endl;
 return 0;
}
int f(int a){
        return (2*a) ;
}
double f(int c){
        return (c/2.0);
}
```

Exercice 3 (optionnel)

Quels sont les affichages produits par ce programme ? Pourquoi ?

?

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

void ecrire(string, int,string = "Nadja",  int = 10);
void ecrire(string, int, int = 10, string = "Dong");

int main(){
    ecrire ("Bonjour", 15, "Ibrahim", 23 );
    ecrire ("Au revoir", 15, "Celeste");
    ecrire ("Salut", 15, 15);
    return 0;
}

void ecrire (string ch1, int i1, string ch2, int i2){
   cout <<"Dans ecrire 1 : " << ch1 <<" "<< ch2 <<" "<< i1 <<" "<< i2 << endl ;
}

void ecrire (string ch1, int i1, int i2, string ch2){
   cout <<"Dans ecrire 2 : " << ch1 <<" "<< ch2 <<" "<< i1 <<" "<< i2 << endl ;
}
```

Que se passerait-il si on ajoutait l'instruction suivante dans le main ?

```
ecrire("au revoir", 15);
```

Partie 2 : Exceptions

Créer un projet vide et y ajouter le programme **exceptions.cpp**.

1. Le programme **exceptions.cpp** utilise un sous-programme **operation** pour réaliser des calculs élémentaires. Le sous-programme **operation** prend en paramètre 2 entiers et un caractère désignant l'opération à effectuer **('+', '-', '*', '/')** et retourne le résultat de l'opération demandée entre les deux entiers. Exécuter le programme.
2. Deux cas d'exécution anormale doivent être prévus. Nous allons les traiter par des exceptions :

- division par 0
- caractère représentant l'opérateur incorrect

1. Modifier maintenant le sous-programme **operation** pour que l'erreur de division par 0 provoque le lancement d'une exception de type entier et de valeur 1. Modifier le main pour que l'exception soit attrapée dans le programme principal et traitée par l'affichage « division par zéro impossible » sur le flot **cerr**. Vérifier à l'exécution.
2. Modifier maintenant le sous-programme **operation** pour lancer en plus une exception de type entier et de valeur 2 si le caractère représentant l'opération demandée est incorrect. Compléter le programme principal pour que l'exception soit attrapée dans le programme principal et traitée par l'affichage « **opération inconnue** ». Vérifier à l'exécution.
3. Et si on ne traite par une exception ? Retirer temporairement – ou mettre en commentaire – le **traitement** (catch) des exceptions dans le programme principal. Tester. Que se passe-t-il ?
4. Et si on oublie tout simplement de placer l'appel des sous-programmes dans une zone protégée (bloc **try**) ? Tester. Que se passe-t-il ?
5. **(optionnel)** Modifier le type de l'exception de la question 3 pour lancer une exception de type **string** (au lieu de **int**) et de valeur « **operation inconnue** » si l'opération demandée n'existe pas. Modifier le programme principal pour que l'exception soit attrapée dans le programme principal. Vérifier.
6. **(optionnel)** Deux en un : Modifier maintenant le sous-programme **operation** pour lancer cette fois-ci des exceptions de type string et de valeurs « **division impossible** » ou « **operation inconnue** » selon le cas. Modifier le programme principal pour que l'exception soit attrapée dans le programme principal. Modifier et vérifier.

Modifié le: lundi 30 octobre 2023, 11:27