Lab1.1

Erstellen Sie ein C++-Programm mit der Bezeichnung `calc`, welches einen einfachen Taschenrechner für ganze Zahlen und den Operatoren +, −, `*` und `/` simuliert. Die beiden Operanden und der Operator werden dem Programm mittels Kommandozeilenargumenten der main()-Funktion übergeben. Beachten Sie, dass sämtliche Fehler (falsche Anzahl von Argumenten, falscher Operator, Operanden können nicht in ganze Zahlen umgewandelt werden, Division durch 0, usw.) vom Programm erkannt werden müssen! Sollte ein Fehler auftreten, terminiert das Programm mit der Fehlermeldung `usage: calc opnd1 op opnd2 (`*`error message`*`)` (siehe unten)!
Bei korrekter Übergabe der Parameter wird die Rechnung samt Ergebnis auf der Konsole ausgegeben. Folgende Beispiele sollen die Funktionalität des Programms demonstrieren:

Beispiele:

```
a)  calc 10 + 20
    10 + 20 = 30
b)  calc 100 / 10
    100 / 10 = 10
c)  calc 10 + 10 + 10
    usage: calc opnd1 op opnd2 (invalid count auf arguments)
d)  calc 10 +
    usage: calc opnd1 op opnd2 (invalid count auf arguments)
e)  calc 10.3 + 10
    usage: calc opnd1 op opnd2 (unable to convert operand to long)
f)  calc Hello + World
    usage: calc opnd1 op opnd2 (unable to convert operand to long)
g)  calc 10 / 0
    usage: calc opnd1 op opnd2 (division by zero)
h)  calc 10 % 2
    usage: calc opnd1 op opnd2 (unknown operator)
```

Gehen Sie bei der Implementierung der Aufgabenstellung folgendermaßen vor:

1. Überprüfung der Anzahl der übergebenen Elemente
   Sollte die Anzahl der übergebenen Elemente ungleich vier (Programmname, Operand1, Operator, Operand2) sein, wird das Programm mit der Ausgabe einer Fehlermeldung auf der Konsole und der Rückgabe eines Fehlercodes (als Konstante definiert) beendet.
   Beachten Sie, dass die Anzahl der Kommandozeilenargumente bzw. die Kommandozeilenargumente selbst als Parameter der `main`()-Funktion zur Verfügung stehen (`int main(int argc, char* argv[])`)! Bei der Angabe eines korrekten Ausdruck `1 + 2` würde der Sachverhalt folgendermaßen aussehen:

   ```
   argc        4
   argv[0]     calc
   argv[1]     1
   argv[2]     +
   argv[3]     2
   ```

2. Umwandlung der Operanden, die in Form von Zeichenketten vorliegen, in ganze Zahlen
   Die Verwendung der Funktion `atoi` ermöglicht keine Feststellung, ob die gesamte umzuwandelnde Zeichenkette auch tatsächlich fehlerfrei umgewandelt werden konnte! Daher ist für die Typkonvertierung einer Zeichenkette (`string`) in eine Variable vom Typ `long` die Funktion `strtol` zu verwenden! Informationen zu den diversen Funktionen bzw. Typkonvertierungen finden Sie in den Links auf der Moodle-Plattform (www.eduvidual.at)!
   Tritt bei der Typkonvertierung ein Fehler auf, wird das Programm mit einer Fehlermeldung auf der Konsole und der Rückgabe eines Fehlercodes (als Konstante definiert) beendet!
3. Fallunterscheidung des Operators
   Der Operator liegt ebenfalls in Form einer Zeichenkette, welche eine Länge von 1 haben muss, damit diese überhaupt korrekt sein kann, vor. Daher muss zuerst mithilfe der `strlen()`-Funktion (`#include <cstring>` erforderlich!) die Länge von `argv[2]` überprüft werden! Sollte die Länge ungleich 1 betragen, wird das Programm mit einer Fehlermeldung auf der Konsole und der Rückgabe eines Fehlercodes (als Konstante definiert) beendet!
   Um die `switch`-Anweisung zur Fallunterscheidung der Rechenoperation verwenden zu können, muss der in `argv[2]` als Zeichenkette enthaltene Operator zuerst in einer Variable eines

ganzzahligen Typs (sinnvollerweise `char`) gespeichert werden. Dies kann durch die Anweisung `char op = argv[2][0]` erfolgen. Erst jetzt kann in den einzelnen `case`-Blöcken der `switch`-Anweisung die Operation ausgeführt und das Ergebnis in einer Variablen gespeichert werden! Beachten Sie, dass es im Fall des Divisionsoperators zu einer Division durch 0 kommen kann! Beachten Sie weiters, dass bei Erreichen des `default`-Blocks kein gültiger Operator erkannt wurde! In beiden Fällen wird das Programm mit einer Fehlermeldung auf der Konsole und der Rückgabe eines Fehlercodes (als Konstante definiert) beendet!

4. Ausgabe der Rechenoperation und des Ergebnisses
   Ist dieser Punkt erreicht, handelt es sich offensichtlich um einen gültigen Ausdruck. Daher ist die Rechenoperation samt Ergbnis in der Form *`opnd1 op opnd2 = result`* in der Konsole auszugeben und das Programm zu beenden.

Hinweise:

- Die Datei mit dem Quellcode ist mit einem Header, welcher zumindest Name, Klasse und Programmname enthält, zu versehen!
- Fügen Sie sinnvolle Kommentare hinzu!
- Verwenden Sie sprechende englische Bezeichner für Variablen uns Konstanten!
- Beachten Sie richtige Einrückungen!
- Auf der Moodle-Plattform (www.eduvidual.at) finden Sie wertvolle Hinweise und Hilfestellungen zu diversen – in diesem Übungsbeispiel enthaltenen – Themen!