Оператор выбора

Напишем простой калькулятор, который будет уметь выполнять два действия — складывать и умножать.

В bash есть несколько способов выполнять арифметические операции:
let
expr
(())

Обратите внимание, что bash умеет выполнять действия только с целыми числами.
Для всех приведённых способов список арифметических операций одинаковый.
+ — сложение
− — вычитание
* — умножение
/ — деление
** — возведение в степень
% — получения остатка от деления

Простые операции в Bash

Начнём с *let*, тут всё очень просто:

```
let a=3+5 ; echo $a
```

let не умеет самостоятельно выводить результат вычисления, только записывать его в переменную.

Далее *expr*. Он, в отличие от *let*, выводит результат сразу на стандартный вывод.

```
expr 3+5
```

Обратите внимание, что знак операции обязательно нужно отделять пробелами.
Мы можем одним действием взять результат математической операции и сразу её вывести. Чтобы взять результат выражения, используются либо бэктики, либо значок доллара и скобки.

Таким образом, чтобы сразу вывести результат работы expr при помощи *echo*, мы можем сделать так:

```
echo $(expr 3 + 5)
```

Давайте теперь попробуем не сложение, а умножение.

```
echo $(expr 3 * 5)
```

Ошибка!

Чтобы bash не пытался интерпретировать звёздочку как шаблон, а просто передал символ звёздочки команде expr, её нужно специальным образом выделить. При помощи значка бэкслеш (это слеш, наклонённый в другую сторону: \).

```
echo $(expr 3 \* 5 )
```

Вот так всё работает.

Последний вариант, конструкция из двойных скобок, может вести себя и так, как *let,* и так, как *expr*.

Мы можем сохранить результат операции в переменную:

```
((a=3+3)) ; echo a
```

Обратите внимание, что в отличие от квадратных скобок тут отделять скобки пробелами от их содержимого необязательно. А можем сразу вывести результат при помощи *echo*.

Оператор выбора

Как выполнять различные действия в зависимости от того, что ввёл пользователь?

Для этих случаев в шелле есть специальный оператор — оператор выбора.
У него очень простой синтаксис:

```
case "$переменная" in
"вариант1" ) команда ;;
"вариант2" ) команда ;;
*) команда
esac
```

Сначала ключевое слово case, затем та переменная, значение которой мы будем сравнивать с различными вариантами. Далее шаблоны для различных случаев — вариант значения переменной и команда, которая будет выполняться. Обратите внимание: звёздочкой обозначается действие по умолчанию, которое будет выполняться, если нам не подходит ни один вариант выше.

Начнём писать наш скрипт.
Сначала спросим у пользователя и считаем два числа. Затем спросим и считаем знак операции.

```
read -p "enter first value:" x
read -p "enter second value" y
read -p "enter action symbol" operation
```

Теперь начнём писать оператор выбора:

```
case $operation in
```

Шаблоны для сложения и умножения с подсчётом и выводом результатов:

```
"+") echo " $x + $y ="  $(expr "$y" + "$x");;
"*") echo " $x * $y =" $(expr  "$x" \* "$y");;
```

В качестве действия по умолчанию давайте выведем сообщение о том, что такой оператор не поддерживается. И не забудем ключевое слово esac.

```
*) echo "unknown operation!"
esac
```

Сохраним наш скрипт и проверим, что он работает: давайте сложим два числа и перемножим два числа.

```
read -p "enter first value:" x
read -p "enter second value" y
read -p "enter action symbol" operation

case $operation in
"+") echo " $x + $y ="  $(expr "$y" + "$x");;
"-") echo "$x - $y ="   $(expr "$x" - "$y");;
"/") if [ $y -eq 0 ]; then
      echo "error: division by zero!";
      else
      echo " $x / $y =" $(expr  $x / $y);
      fi;;
"*") echo " $x * $y =" $(expr  $x \* $y);;
*) echo "unknown operation!"
esac
```