Быстрый оператор apply вместо функции Eval

admin
December 2021 edited December 2021

Я уже писал про использование функции Eval. Главная её практическая польза - возможность использования (динамического встраивания) в алгоритме некоторого выражения переданного в виде объекта из другого места. Это работает, но замедляет выполнение примерно на 65%. Это происходит потому, что Eval это функция, и на её вызов уходит некоторое время. Поэтому я сделал специальный оператор apply (и apply$), который, как и Eval, выполняет выражение из переменной, но без возможности возврата значения. Переделал ранее показанный пример с Eval под apply.

```
int[] arr=new int[]{5, 6, 7};
object someExpr=GetExpr(); //получаем выражение
ForAllElements(arr, someExpr); //передаём в функцию массив, который нужно изменить, и выражение
WriteLine(Join(' ', arr)); //15 16 17 - к каждому элементу прибавилось 10
ReadKey();

int[] ForAllElements(int[] arr, object e){
    //эта функция для каждого индекса элемента массива выполняет выражение, которое меняет значение элемента
    foreach(int index in Range(arr.Length()))
        apply$ e; //apply$ выполняет выражение arr[index]+=10 в текущем контексте
        //можно и apply использовать, но в цикле будет работать медленнее
}

object GetExpr(){
    //эта функция создаёт выражение с массивом и индексом
    int[] arr;
    int index;
    return Expr(arr[index]+=10); //суть выражения - прибавление к элементу числа 10
}
ReadKey();
```

Отличаются apply и apply$ тем, что последний кеширует значение переменной с выражением (не результат вычисления, а только ссылку на выражение), а apply каждый раз считывает его заново. Поэтому apply$ работает быстрее, когда применяется в цикле. Если apply замедляет работу на 29%, то apply$ всего на 3%. Но нужно понимать, что apply$ всегда выполняет то выражение, которое выполнил в первый раз, и изменение выражения в переменной-аргументе не повлияет на его работу. Вне циклов лучше apply потому, что при одиночном вызове кэширование является лишним.
Я называю apply оператором, но это не такой оператор как + и другие, а более низкоуровневый, как if, for, goto, throw. Это всё statements (инструкции), которые работают на уровне логической строки кода, и их нельзя использовать в выражениях.