Оуэн постоянно старается улучшить свою игру...

Хорошие гейм-мастера стремятся создать у игроков наилучшие впечатления от игры. Группа Оуэна собирается начать новую кампанию с новыми персонажами, и Оуэн думает, что с небольшими изменениями в формуле определения характеристик игра станет более интересной.

Когда игроки заполняют свои листы персонажей в начале игры, они выполняют следующие действия для вычисления каждой из начальных характеристик своего персонажа:

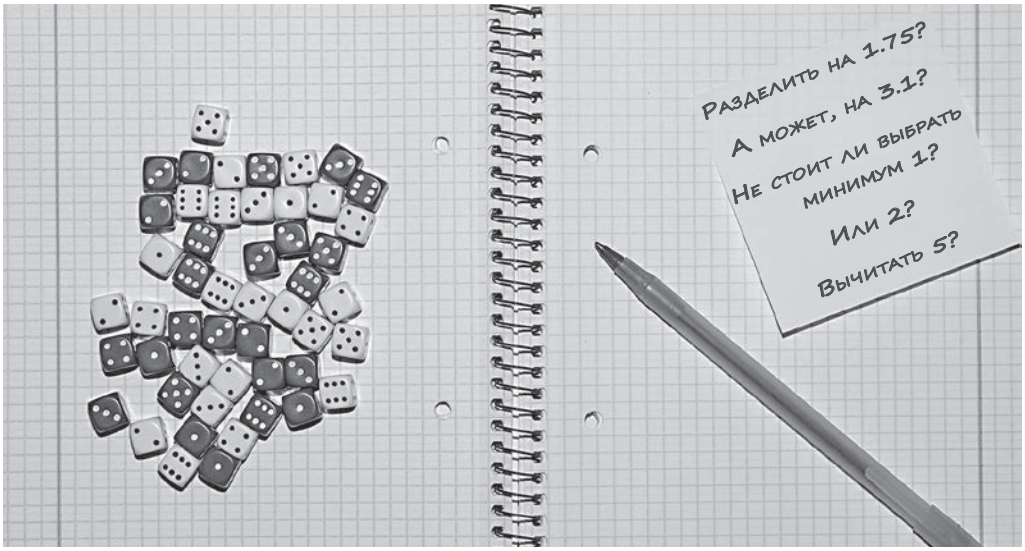
ФОРМУЛА ОПРЕДЕЛЕНИЯ ХАРАКТЕРИСТИК

* Начните с броска 4d6, чтобы получить число от 4 до 24.
* Разделите результат броска на 1.75.
* Прибавьте 2 к результату деления.
* Округлите до ближайшего целого.
* Если результат слишком мал, используйте минимальное значение 3.

«Бросок 4d6» означает, что вы бросаете четыре обычных шестигранных кубика и складываете результаты.

...но процесс проб и ошибок может занимать много времени

Оуэн экспериментировал с различными вариантами настройки вычисления характеристик. Он уверен, что формула в целом хороша, но ему хотелось бы иметь возможность экспериментировать с числами.



Оуэну нравится общая формула: бросить 4d6, разделить, вычесть, округлить, использовать минимальное значение... но он не уверен в правильности конкретных чисел.

Мне кажется, что 1.75 маловато для деления результата броска... И возможно, к результату лучше прибавить 3, а не 4. **Наверняка должен быть более удобный способ** проверки всех этих идей!

МОЗГОВОЙ ШТУРМ

Что мы можем сделать, чтобы помочь Оуэну в поиске оптимальной комбинации значений для обновленной формулы характеристик?

Поможем Оуэну в экспериментах с характеристиками

В следующем проекте мы построим консольное приложение .NET Core, при помощи которого Оуэн сможет протестировать свою формулу вычисления характеристик с разными значениями и проверить, как они влияют на результат. Формула получает **четыре входных значения**: *начальный бросок 4d6*; *делитель*, на который делится результат; *приращение*, которое прибавляется к результату деления, и *минимум*, который используется, если результат окажется слишком маленьким.

Страница из книги гейм-мастера с формулой вычисления характеристик.

Оуэн вводит четыре входных значения в приложении, которое будет вычислять характеристики по этим данным. Вероятно, он захочет протестировать набор разных значений, поэтому для удобства приложение будет снова и снова запрашивать входные данные, пока приложение не будет завершено, отслеживать значения, введенные при каждой итерации, и использовать предыдущие значения **по умолчанию** при следующей итерации.

Вот что должен видеть Оуэн при запуске приложения:

```
C:\Users\public\source\repos\AbilityScoreTester\AbilityScoreTester\bin\Debug\netcoreapp3.1\Ab
Starting 4d6 roll [14]:
   using default value 14
Divide by [1.75]:
   using default value 1.75
Add amount [2]:
   using default value 2
Minimum [3]:
   using default value 3
Calculated ability score: 10
Press Q to quit, any other key to continue
Starting 4d6 roll [14]:
   using default value 14
Divide by [1.75]: 2.15
   using value 2.15
Add amount [2]: 5
   using value 5
Minimum [3]: 2
   using value 2
Calculated ability score: 11
Press Q to quit, any other key to continue
Starting 4d6 roll [14]: 21
   using value 21
Divide by [2.15]:
   using default value 2.15
Add amount [5]:
   using default value 5
Minimum [2]:
   using default value 2
Calculated ability score: 14
Press Q to quit, any other key to continue
```

Приложение запрашивает различные значения, используемые для вычисления характеристик. Значения по умолчанию выводятся в квадратных скобках (например, [14] или [1.75]). Оуэн может ввести значение или просто нажать Enter, чтобы подтвердить значение по умолчанию.

Здесь Оуэн опробует новые значения: результат броска делится на 2.15 (вместо 1.75), результат деления увеличивается на 5 (вместо 2), и при вычислении используется минимальное значение 2 (вместо 3). С исходным броском 14 будет получено значение 11.

Теперь Оуэн хочет проверить те же значения при другом исходном броске 4d6, поэтому он вводит 21 на первый запрос и нажимает Enter, чтобы подтвердить значения по умолчанию, сохраненные приложением при предыдущей итерации. На этот раз будет получено значение 14.

ФОРМУЛА ОПРЕДЕЛЕНИЯ ХАРАКТЕРИСТИК

* Начните с броска 4d6, чтобы получить число от 4 до 24.
* Разделите результат броска на 1.75.
* Прибавьте 2 к результату деления.
* Округлите до ближайшего целого.
* Если результат слишком мал, используйте минимальное значение 3.

Этот проект больше предыдущего консольного приложения, которое вы построили, поэтому мы рассмотрим его за несколько этапов. Сначала мы разберемся в коде вычисления характеристики, затем будет написан остальной код приложения, и наконец, займемся диагностикой ошибок в коде. *Итак, за дело!*

Возьми в руку карандаш

Мы построили класс, который поможет Оуэну в вычислении характеристик. Чтобы использовать его, необходимо задать значения его полей Starting4D6Roll, DivideBy, AddAmount и Minimum (или оставить полям значения, заданные при объявлении) и вызвать метод CalculateAbilityScore. К сожалению, **в одной строке кода допущена ошибка**. Обведите строку с ошибкой и напишите, что с ней не так.

```
class AbilityScoreCalculator
{
    public int RollResult = 14;
    public double DivideBy = 1.75;
    public int AddAmount = 2;
    public int Minimum = 3;
    public int Score;

    public void CalculateAbilityScore()
    {
        // Результат броска делится на значение поля DivideBy
        double divided = RollResult / DivideBy;

        // AddAmount прибавляется к результату деления
        int added = AddAmount += divided;

        // Если результат слишком мал, использовать значение Minimum
        if (added < Minimum)
        {
            Score = Minimum;
        } else
        {
            Score = added;
        }
    }
}
```

Эти поля инициализируются значениями из формулы вычисления характеристик. Приложение использует их при выводе значений по умолчанию.

Удастся ли вам обнаружить проблему, не вводя класс в IDE? Найдете ли вы строку, из-за которой компилятор выдает сообщение об ошибке?

Подсказка: сравните комментарии в коде с формулой вычисления характеристик на странице из книги Оуэна. Какая часть формулы отсутствует в комментариях?

После того как вы **пометите строку кода с проблемой**, запишите, какие проблемы вы в ней обнаружили.

.....

.....

Использование компилятора C# для поиска проблемной строки кода

Создайте проект консольного приложения .NET Core Console App с именем AbilityScoreTester. Затем **добавьте класс AbilityScoreCalculator** с кодом из упражнения «Возьми в руку карандаш». Если код был введен правильно, вы получите ошибку компилятора C#:

```
AddAmount += divided;
```

```
(field) int AbilityScoreCalculator.AddAmount

CS0266: Cannot implicitly convert type 'double' to 'int'. An explicit conversion exists (are you missing a cast?)

Show potential fixes (Alt+Enter or Ctrl+.)
```

Ошибка компилятора C# буквально напоминает о том, что вы могли пропустить приведение типа.

Каждый раз, когда компилятор C# выдает сообщение об ошибке, тщательно прочитайте его. Обычно в нем присутствует подсказка, которая поможет обнаружить проблему. В данном случае причина точно обозначена: компилятор не может преобразовать double в int без приведения типа. Переменная `divided` объявлена с типом double, но C# не позволит добавить ее к полю int (такому, как AddAmount), потому что не знает, как преобразовать ее.

Компилятор дает чрезвычайно ценную подсказку о том, что вы должны выполнить приведение типа double-переменной `divided`, прежде чем прибавлять ее к int-полю AddAmount.

Добавим приведение типа, чтобы класс AbilityScoreCalculator компилировался...

Теперь мы знаем, в чем заключается суть проблемы, и можем добавить **приведение типа** для исправления проблемной строки кода в AbilityScoreCalculator. Ошибка «Не удается неявно преобразовать тип» выдается следующей строкой:

```
int added = AddAmount += divided;
```

Ошибка возникает из-за того, что команда **`AddAmount += divided`** ***возвращает значение double***, которое не может быть присвоено int-переменной added.

Проблему можно решить **приведением `divided`** к **`int`**, чтобы при прибавлении к AddAmount было возвращено другое значение int. Замените в этой строке кода `divided` на `(int)divided`:

```
int added = AddAmount += (int)divided;
```

Преобразуйте!

Приведение также добавляет отсутствующую часть формулы Оуэна:

**Округлите до ближайшего целого.*

Когда вы приводите double к int, C# округляет результат — так что, например, (int)19.7431D дает 19. Добавляя это приведение, вы также добавляете пункт формулы, отсутствующий в классе.

...но ошибка все равно осталась!

Работа еще не закончена! Ошибка компилятора исправлена, так что проект успешно строится. Но хотя компилятор C# не протестует, ***проблема все еще осталась***. Удастся ли вам найти ошибку в следующей строке кода?

Похоже, заполнять ответ во врезке «Возьми в руку карандаш» еще рано!