Каждый файл, использующийся в качестве шаблона для генератора заданий, должен соблюдать определенный набор правил. Каждый шаблон должен состоять из пяти блоков: “ХРАНИЛИЩЕ_ОБЪЕКТОВ”, “ШАБЛОННЫЙ_ВИД”, “РЕШЕНИЕ”, “СЛУЖЕБНОЕ”, “ТЕСТОВЫЕ_ДАННЫЕ”.

Описание грамматики блока:

```
блок = “----{-}” конец_строки название_блока конец_строки содержимое_блока
конец_строки = ‘\n’
название_блока = “ХРАНИЛИЩЕ_ОБЪЕКТОВ” | “ШАБЛОННЫЙ_ВИД” | “РЕШЕНИЕ” | “СЛУЖЕБНОЕ” | “ТЕСТОВЫЕ_ДАННЫЕ”
содержимое_блока = “осмысленный набор символов”
```

Каждый блок должен быть описан в шаблоне единожды. Ниже представлен пример формата шаблона:

```
----
ХРАНИЛИЩЕ_ОБЪЕКТОВ
содержимое_блока
----
ШАБЛОННЫЙ_ВИД
содержимое_блока
----
РЕШЕНИЕ
содержимое_блока
----
СЛУЖЕБНОЕ
содержимое_блока
----
ТЕСТОВЫЕ_ДАННЫЕ
содержимое_блока
```

Блок “ХРАНИЛИЩЕ_ОБЪЕКТОВ” содержит объекты-параметры, которые составляют основу для генерации заданий и эталонных решений. Форма объекта_шаблона:

```
объект_параметр = “имя объекта-параметра” “:” значение_объекта_параметра {“,” значение_объекта_параметра} “;”
значение_объекта_параметра = “набор символов юникода” | функция
```

```
функция = “#имя_функции({аргументы функции [|]})”
```

Объект-параметр может содержать от одного до бесконечности некоторых значений, в дальнейшем случайным образом будет выбрано одно значение, которое и будет подставляться в местах вызова объекта-параметра. Для использования объектов_параметров в нужной части шаблона, объект-параметр необходимо поместить в скобки следующим образом: “(имя_объекта_параметра)”. Замечание: пробелов и других символов табуляции находиться между скобками и именем объекта-параметра не должно.

Генератор предоставляет некоторый изначальный набор функций, а также функции написанные пользователями на языке Lua (расширение стандартного набора функций с помощью Lua будет реализован в дальнейшем). На данный момент генератор предоставляет три основных функции и одну второстепенную:

- #rnd (нижняя_граница | верхняя_граница | тип {| количество}) - генерирует случайное число в границе [нижняя_граница, верхняя_граница) типа “тип” (double или int). Если необходимо сгенерировать сразу несколько значений используется дополнительный параметр “количество”, в таком случае сгенерированные числа будут разделены запятыми;
- #genAE(сложность {| граница | тип | количество переменных}) - генерирует выражение некоторой сложности (сложность указывает количество знаков и функций (например log, sin и т.д.)), граница указывает диапазон генерации констант [-граница, граница), тип указывает на тип генерируемых констант (double или int), количество переменных, указывает на зависимвость выражения от количества переменных (от 1 до бесконечности);
- #genAEcode() - возвращает код для вычисления последнего сгенерированного выражения (вместе с проверками на возможность вычисления выражения) на языке C++.
- #lua(код_на_луа) - данная функция является второстепенной и предназначена для выполнения код_на_луа (изначально предполагалось использовать луа лишь для вычисления мат. выражений, но в дальнейшем данный язые будет использовать для расширения предоставляемого функционала генератора). Все системные функции (кроме получения времени и даты) на lua которые обращаются к OS запрещены в целях безопасности.

Пример описание блока “ХРАНИЛИЩЕ_ОБЪЕКТОВ”:

```
----
ХРАНИЛИЩЕ_ОБЪЕКТОВ
предложение : некоторое предложение содержащее цифры 123 и буквы, второе значение параметра; // из двух значений выберется лишь одно
```

```
предложение2 : вот содержимое первого параметра (предложение);
// конечный вид предложение2 = вот содержимое первого параметра некоторое предложение содержащее цифры 123 и буквы (если бы выбралось первое значение)

Число : #rnd(10 | 30 | int) , 123, 321 , 451; // если случайно выберется первое значение будет сгенерировано целое число в диапазоне, иначе будет выбраны другие числа

Зависимое число: #rnd( (Число) | 80 | int); // в данном случае вместо (Число) подставится содержимое объекта-параметра Число

Выражение : #genAE(5); // сгенерируется мат. выражение сложность 5 со стандартными значениями, то есть, константы будут в диапазоне [-100, 100), типа double, мат. выражение будет зависеть от одного параметра.

Код для вычисления выражения : #genAEcode();

Выражение с настройками : #genAE(5 | 50 | int | 2); // сгенерируется мат. выражение сложность 5, с константами [-50, 50), типа int, зависимое от двух аргументов
```

Примечание : все ключевые символы могут экранироваться с помощью “\”.

Блоки “ШАБЛОННЫЙ_ВИД”, “РЕШЕНИЕ” довольно похожи. По сути являющиеся текстом со вставками объектов-параметров :

- “ШАБЛОННЫЙ_ВИД” - задание которое будет созданно и предоставлено обучаемому,
- “РЕШЕНИЕ” - эталонное решение которое будет компилироваться и в дальнейшем использоваться для проверки присылаемых работ.

Пример использования данных блоков:

```
----
ХРАНИЛИЩЕ_ОБЪЕКТОВ
выражен1 : #genAE(7);
код1 : #getAEcode();

x_нач : #rnd( 0.0 | 5.0  | double);
x_кон : #rnd( 7.0 | 10.0 | double);
    N : #rnd( 10  | 20   | int);
----
ШАБЛОННЫЙ_ВИД
Вычислите и выведите на экран значения функции y = f(x)
в точках x0=x_нач, x1=x0+h, x2=x1+h, … xN =x_кон, где h =( x_кон – x_нач ) / N .
Иными словами: затабулируйте функцию y = f(x) на отрезке[x_нач ; x_кон ] с шагом h.

f(x)  = (выражен1)
x_нач = (x_нач)
x_кон = (x_кон)
    N = (N)
----
РЕШЕНИЕ
```

```
#include <cmath>
#include <iostream>

using namespace std;

int main()
{
    double y = 0.0;
    double h = ((x_кон) - (x_нач)) / (N);
    for (double x1 = (x_нач); x1 < (x_кон); x1 += h)
    {
        (код1)
        cout << x1 << "  |  " << y << endl;
    }

    system("pause");

    return 0;

}
```

В результате получится:

```
Задание:
Вычислите и выведите на экран значения функции y = f(x)
в точках x0=x_нач, x1=x0+h, x2=x1+h, : xN =x_кон, где h =( x_кон - x_нач ) / N .
Иными словами: затабулируйте функцию y = f(x) на отрезке[x_нач ; x_кон ] с шагом h.

f(x)  = 45.13 * pow((8.89 * ((97.71 * ((-38.49 + x1) * x1)) / x1)), x1)
x_нач = 4.99
x_кон = 8.54
    N = 12


Исходный код эталонного решения:
#include <cmath>
#include <iostream>

using namespace std;

int main()
{
    double y = 0.0;
    double h = (8.54 - 4.99) / 12;
    for (double x1 = 4.99; x1 < 8.54; x1 += h)
    {
        if (x1 != 0 )
             y = 45.13 * pow((8.89 * ((97.71 * ((-38.49 + x1) * x1)) / x1)), x1);
        else
             y = 0.0;
        cout << x1 << "  |  " << y << endl;
    }

    system("pause");

    return 0;

}
```

В шаблоне присутствует еще один блок “СЛУЖЕБНОЕ”, данный блок содержит определенные параметры, отвечающие за настройку некоторых функций. На данный момент

имеется лишь две настройки:

- знаки_арифм : значение{, значение}; – отвечает за символы, которые будут использоваться во время генерации арифметических выражений;
- функции_арифм : значение{, значение}; – отвечает за функции, которые будут использоваться во время генерации арифметических выражений.

Пример использования блока:

```
----
СЛУЖЕБНОЕ
знаки_арифм : +, -, *, /, ^;
функции_арифм : sin, cos, sqrt, tan;
```

Последний блок это “ТЕСТОВЫЕ_ДАННЫЕ”, данные которые будет использовать тестирующий модуль при сравнении присылаемого решения и эталонного решения. Тестовые данные могут заранее заготовленные, а также случайно сгенерированные. Ниже представлен пример использования блока:

```
----
ХРАНИЛИЩЕ_ОБЪЕКТОВ
x_1 : #rnd(5.0 | 10.0 | double);
----
...
----
ТЕСТОВЫЕ_ДАННЫЕ
Тест1 : #rnd(100 | 200 | int | 1000); // генерация 1000 случайных челых чисел в диапазоне
Тест2 : 1,2,3,4,5,6,7,8,9,10; // заранее заготовленные данные
Тест3 : #rnd((x_1) | 20.0 | double | 10); // генерации с зависимым параметром
```

Пример готового шаблона:

```
----
ХРАНИЛИЩЕ_ОБЪЕКТОВ
выражен : #genAE(5);
    код : #getAEcode();

a : #rnd(0.1  | 0.35 | double);
b : #rnd(0.75 | 1.1  | double);
N : #rnd(10   | 40   | int);

----
ШАБЛОННЫЙ_ВИД
Для x, изменяющегося от a до b (интервал [a ; b] целиком лежит внутри интервала)
с шагом h = (b-a)/N, вычислить функцию y=f(x), используя ее разложение в степенной ряд.

f(x) = (выражен)
a = (a)
b = (b)
N = (N)
----
РЕШЕНИЕ
```

```
#include <cmath>
#include <iostream>

using namespace std;

int main()
{
    double y = 0.0;
    double h = ((b) - (a)) / (N);
    for (double x1 = (a) ; x1 <= (b); x1 += h)
    {
        (код)
        cout << y << endl;
    }
    system("pause");
    return 0;
}
----
СЛУЖЕБНОЕ
знаки_арифм : +, -, *, /;
функции_арифм : sin, log, cos, tan;

----
ТЕСТОВЫЕ_ДАННЫЕ
тест1 : #rnd(1.0 | 20.0 | double | 10);
тест2 : 1,2,3,4,5,6,7,8,9,10;
---
```