Техническая документация к SimpleRNG

Обзор:
Класс SimpleRNG реализует линейный конгруэнтный генератор псевдослучайных чисел по формуле

$x[n+1]=(a\times x[n]+c)\%m$
где

1. $m>1(модуль)$
2. $0<a<1(множитель)$
3. $0<c<m(приращение)$
4. $x[0](начальное\,состояние/seed)$

Класс предоставляет итератор, совместмый с STL, для последовательной генерации чисел с автоматическим определением циклов.

Файл SimpleRNG.h

1. Класс SimpleRNG

Приватные поля:

1. modulus, multiplier, increment: параметры генератора (m,a,c)
2. currentState: текущее состояние генератора x[n]
3. initialState: начальное состояние генератора x[0], заданное через reset()

Конструктор:
SimpleRNG(double m, double a, double c)

Как работает:

1. Инициализирует параметры генератора
2. Проверяет корректность входных параметров (выбрасывает исключения)
3. Устанавливает начальное состояние в 0 (требует вызова reset())

Методы управления состоянием:
void reset(double seed)

Как работает:

1. Сохраняет seed как initialState
2. Устанавливает currentState в seed
3. Позволяет начать генерацию с нового начального значения

void reset()

Как работает:

1. Восстанавливает currentState из сохраненного initialState

2. Позволяет повторно генерировать ту же последовательность

double next()

Как работает:
1. Вычисляет следующее значение по формуле:
   $(multiplier \times currenrState + increment)\,\%\,modulus$
2. Использует std::fmod для работы с числами с плавающей точкой
3. Обновляет currentState и возвращает результат

Методы итераторов:
Iterator begin()

Как работает:
1. Создает итератор, связанный с текущим объектом SimpleRNG
2. Итератор начинает с текущего состояния генератора
3. Использует epsilon по умолчанию (0.05)

Iterator end(double eps = 0.05)

Как работает:
1. Создает конечный итератор-маркер
2. eps определяет точность сравнения для обнаружения циклов
3. Используется как критерий остановки в циклах

Константные версии begin() и end()

Как работает
1. Предоставляют константный интерфейс для итерации
2. Внутренне используют const_cast для вызова неконстантных методов
3. Обеспечивают совместимость с range-based for для константных объектов

2. Вложенный класс Iterator

Приватные поля:
1. rngPtr: указатель на связанный генератор
2. currentVal: текущее значение, возвращаемое итератором
3. epsilon: точность сравнения чисел с плавающей точкой
4. isEndFlag: флаг конечного итератора
5. cycleFound: флаг обнаружения цикла
6. stateHistory: вектор предыдущих состояний для отслеживания циклов

Определения типов:
using iterator_category = std::input_iterator_tag;
using value_type = double;

using difference_type = std::ptrdiff_t;
using pointer = double*;
using reference = double&;

Конструкторы:
Iterator()

Как работает:
Создает конечный итератор-маркер с флагом isEndFlag = true
Iterator(SimpleRNG* generator, double eps, bool endMarker)

Как работает:

1. Если endMarker = false и генератор существует:
    - Сохраняет начальное состояние в currentVal
    - Добавляет его в историю состояний
1. Если endMarker = true: создает конечный итератор

Методы:
bool stateInHistory(double state, double eps) const

Как работает:

1. Проверяет, встречалось ли состояние state ранее
2. Сравнивает с учетом точности eps $|(state - oldState)| < eps$
3. Возвращает true при обнаружении похожего состояния

double operator*() const

Как работает:
Возвращает текущее значение currentVal

Iterator& operator++() (префиксный)

Как работает:

1. Генерирует следующее значение через rngPtr→next()
2. Проверяет, не встречалось ли это значение ранее
3. Если значение новое:
    - Добавляет в историю
    - Обновляет currentVal
1. Если значение повторяется:
    - Устанавливает флаги окончания (isEndFlag, cycleFound)

Iterator operator++(int) (постфиксный)

Как работает:

1. Сохраняет текущее состояние итератора

2. Вызывает префиксный инкремент
3. Возвращает сохраненное состояние

bool operator==(const Iterator& other) const

Как работает:
1. Оба итератора конечные - равны
2. Один конечный, другой нет - не равны
3. Обнаружен цикл в любом из итераторов - равны
4. Иначе, сравнивает currentVal с точностью epsilon

bool operator!=(const Iterator& other) const

Как работает:
Отрицание оператора ==

Файл SimpleRNG.cpp

Конструктор SimpleRNG:
Как сделано:
1. Использует список инициализации для установки значений
2. Выполняет три проверки параметров
   $m > 1$
   $0 < a < 1$
   $0 < c < m$
3. При нарушении условий выбрасывает std::invalid_argument

Метод next()
Как сделано:
1. Вычисляет $multiplier \times currentState + increment$
2. Применяет модуль через std::fmod()
3. Обновляет currentState и возвращает результат

Методы итераторов
Как сделано:
1. begin(): создает итератор с текущим объектом
2. end(): создает конечный итератор с указанным epsilon
3. Константные версии используют const_cast для обхода ограничений

Файл IteratorMethods.cpp

Как работает:
При каждом инкременте итератор
1. Генерирует новое значение
2. Проверяет его наличие в истории

3. При обнаружении похожего значения в пределах epsilon помечает цикл найденным

Сравнение итераторов
Как работает:

1. Конечные итераторы равны друг другу
2. Итератор с обнаруженным циклом равен любому итератору
3. Сравнение значений происходит с учетом epsilon