Компьютерная Безопасность
МИЕМ НИУ ВШЭ

Работа: Мельникова Льва СКБ 161
Вариант 19

# Сравнение генераторов псевдо-случайный чисел

Были исследованы следующие алгоритмы:

1. xorshift
2. Линейный конгруентный метод
3. Обертка на rand из cstdlib

Были вычислены минимумы, максимумы, среднее, отклонения, коэфициенты корреляции на выборках составленных этими алгоритмами.

Описание выборки
Диапазон: 5000
Количество элементов: 30000
Количество степеней свободы: 10
Квантиль принятия по Хи-Квадрат:90%

Исследовались значения на 10 выборках.

Результаты stdcrand
min|max: 0|4999
mean|deviation|variation: 2514.54|1442.07|0.573492
chi|percentile: 43.209|90
min|max: 0|4999
mean|deviation|variation: 2503.62|1439.38|0.57492
chi|percentile: 34.8393|55
min|max: 0|4999
mean|deviation|variation: 2507.5|1440.18|0.57435
chi|percentile: 33.0473|45
min|max: 0|4999
mean|deviation|variation: 2490.41|1443.31|0.579546
chi|percentile: 36.545|65
min|max: 0|4999
mean|deviation|variation: 2482.46|1446.07|0.582515
chi|percentile: 39.9843|75
min|max: 0|4999
mean|deviation|variation: 2510.07|1443.69|0.575161
chi|percentile: 30.9567|35
min|max: 0|4999
mean|deviation|variation: 2495.79|1443.76|0.578479
chi|percentile: 23.812|20
min|max: 0|4999
mean|deviation|variation: 2493.05|1442.69|0.578687
chi|percentile: 29.6943|30
min|max: 0|4999
mean|deviation|variation: 2489.39|1455.01|0.584486
chi|percentile: 42.7027|85
min|max: 0|4999
mean|deviation|variation: 2490.96|1439.92|0.578057
chi|percentile: 38.3113|70
CHI MEAN:35.3102
PERCENTILE MEAN:57

Результаты xorshift
min|max: 0|4999
mean|deviation|variation: 2496.99|1448.5|0.580098
chi|percentile: 41.284|80
min|max: 0|4999
mean|deviation|variation: 2506.05|1437.81|0.573737
chi|percentile: 32.814|45
min|max: 0|4999
mean|deviation|variation: 2485.62|1442.1|0.580179
chi|percentile: 48.6527|95
min|max: 0|4999
mean|deviation|variation: 2491.33|1440.59|0.578241
chi|percentile: 20.053|10
min|max: 0|4999
mean|deviation|variation: 2493.47|1443.32|0.57884
chi|percentile: 37.224|65
min|max: 0|4999
mean|deviation|variation: 2483.11|1442.72|0.581013
chi|percentile: 39.492|75
min|max: 0|4999
mean|deviation|variation: 2499.99|1446.81|0.578728
chi|percentile: 34.1837|50
min|max: 0|4999
mean|deviation|variation: 2496.3|1441.71|0.577541
chi|percentile: 29.2043|30
min|max: 0|4999
mean|deviation|variation: 2500.32|1441.9|0.576684
chi|percentile: 37.7|70
min|max: 0|4999
mean|deviation|variation: 2501.15|1444.21|0.577418
chi|percentile: 23.532|20
CHI MEAN:34.414
PERCENTILE MEAN:54

Результаты линейного конгруэнтного метода
min|max: 0|4999
mean|deviation|variation: 2497.69|1449.97|0.580523
chi|percentile: 28.2057|25
min|max: 0|4999
mean|deviation|variation: 2505.3|1443.79|0.576296
chi|percentile: 31.9297|40
min|max: 0|4999
mean|deviation|variation: 2497.88|1438.02|0.575697
chi|percentile: 43.0247|85
min|max: 0|4999
mean|deviation|variation: 2507.76|1448.24|0.577502
chi|percentile: 33.983|50
min|max: 0|4999
mean|deviation|variation: 2494.65|1441.04|0.577651
chi|percentile: 28.0027|25
min|max: 0|4999
mean|deviation|variation: 2491.7|1440.89|0.578277
chi|percentile: 32.702|45
min|max: 0|4999
mean|deviation|variation: 2503.81|1448.6|0.578556
chi|percentile: 29.4307|30
min|max: 0|4999
mean|deviation|variation: 2501.11|1446.16|0.578207
chi|percentile: 31.4467|40
min|max: 0|4999
mean|deviation|variation: 2511.24|1445.1|0.575453
chi|percentile: 34.956|55
min|max: 0|4999
mean|deviation|variation: 2492.96|1446.73|0.580325
chi|percentile: 33.423|50
CHI MEAN:32.7104
PERCENTILE MEAN:44.5

Выводы:

Для всех трех генераторов, есть выборки на которых критерий Хи-Квадрат принимает значения на уровне близкому к 50% квантиле.

Можно говорить, что эти методы случайны

Код Xorshift

```
uint32_t generate(uint32_t& state, uint32_t range)
{
    uint32_t result = state;

    // XOR shift
    result ^= result << 13;
    result ^= result >> 17;
    result ^= result << 5;

    state = result;

    return result % range;
}
```

Код линейного конгруентного метода

```
    uint32_t generate(uint32_t &state, uint32_t range) {
        // constants suggested by glibc
        constexpr std::size_t mod = 2147483648;
        constexpr std::size_t koef = 1103515245;
        constexpr std::size_t addition = 12345;

        state = (state * koef + addition) % mod;

        return state % range;
    }
```

Код обертки над rand

```
uint32_t stdcrand(uint32_t& state, uint32_t range) {
    return rand() % range;
}
```

Код исследования выборки

```
struct result {
    double mean;
    uint32_t max;
    uint32_t min;
    double deviation;
    double variation;
    double chi_level;
    double accepted_chi_level;
    bool passed;
};

std::vector<result> test(prng_func_ptr function, std::size_t range,
std::size_t elements, std::size_t repeat,
                         double chi_level_accepted, std::size_t dof) {

    std::vector<result> results;
    double dof_koef = static_cast<double>(dof) / range;
    for (std::size_t j = 0; j < repeat; j++) {
        result res;
        std::map<uint32_t, uint32_t> map;
        uint32_t state = j + 100;
        res.deviation = 0;
        res.mean = 0;
        res.variation = 0;
        double predicted_mean = static_cast<double>(range - 1) / 2;
        for (std::size_t i = 0; i < elements; i++) {
            auto value = function(state, range);
            res.max = std::max(value, res.max);
            res.min = std::min(value, res.min);
            res.deviation += (predicted_mean - value) * (predicted_mean
- value);
```

```
            res.mean += value;
            value = value*dof_koef;
            if (map.find(value) == map.end())
                map[value] = 0;
            map[value] += 1;
        }
        res.mean = static_cast<double>(res.mean) / elements;
        res.deviation /= elements;
        res.deviation = std::sqrt(res.deviation);
        res.variation = res.deviation / res.mean;
        res.accepted_chi_level = chi_level_accepted;

        res.chi_level = 0;
        for (std::size_t i = 0; i < dof; i++) {
            double got = 0;
            if (map.find(i) != map.end())
                got = map[i];
            double expected = static_cast<double>(elements) / dof;
            res.chi_level += (got - expected) * (got - expected) /
expected;
        }
        res.passed = false;
        if (res.accepted_chi_level >= res.chi_level)
            res.passed = true;

        results.push_back(res);
    }
    return results;
}
```

Весь код проекта можно посмотреть здесь:
https://github.com/MarkGrin/sortringHomework