Программирование, лекция 6

Кафедра ИУ7 МГТУ им. Н. Э. Баумана, 2021 год

Псевдослучайные числа

Pseudo-random numbers (PRN)

Вырабатываемая алгоритмически последовательность чисел, обладающих свойствами случайных чисел и используемых взамен последних при решении на ЭВМ ряда классов задач

Генератор псевдослучайных чисел (ГПСЧ,, PRNG) — алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются заданному распределению (обычно равномерному).

Требования к ГПСЧ

Источники случайных чисел - физические шумы (использовать сложно, медленно, дорого).

Недостатки ГПСЧ:

- повторяемость (периодичность) последовательности;
- зависимость значений

Требования к ГПСЧ:

- длинный период;
- эффективность;
- воспроизводимость.

Модуль random

Реализует генерацию псевдослучайных чисел различных распределений.

Функции состояния:

- seed(a=None, version=2)
- getstate()
- setstate(state)

Функция генерации последовательности байтов:

- randbytes(n) (с 3.9)

Генерация чисел и последовательностей

Числовые функции:

- **randrange(stop), randrange(start, stop[, step])**
- **randint(a, b)** (алиас для randrange(a, b+1))
- getrandbits(k)

Функции последовательностей:

- choice(seq)
- choices(population, weights=None, *, cum_weights=None, k=1)
- shuffle(x[, random])
- sample(population, k, counts=None)

Распределения

- **random()**
- **uniform(a, b)**
- triangular(low, high, mode)
- betavariate(alpha, beta)
- expovariate(lambd)
- gammavariate(alpha, beta)
- gauss(mu, sigma)
- lognormvariate(mu, sigma)
- normalvariate(mu, sigma)
- vonmisesvariate(mu, kappa)
- paretovariate(alpha)
- weibullvariate(alpha, beta)

Создание списков

```
a = [] # пустой список
a = [0] * 10     # список фиксированного размера, инициализированный
                 #начальными значениями
# списковые включения (list comprehension)
a = [i*i for i in range(10)]
a = [i for i in range(10) if i % 2 == 0]
```

Способы ввода списков

```
# 1. быстро, плохо:

a = list(map(int, input('Введите массив (в одну строку через пробел): ').split()))


# 2. по элементам с указанием размера:

n = int(input('Введите размер массива:'))

a = [0] * n

for i in range(n):

    a[i] = int(input('Введите {}-й элемент: ').format(i+1))
```

Способы ввода списков

```
# по элементам без ввода размера:
a = []
i = 0
while True:
    i += 1
    el = input('Введите {}-й элемент: ').format(i)
    if el:
        a.append(int(el))
    else:
        break
```

Вывод списка

```
print(a) # плохо. подходит только для отладки


# лучше - с форматированием и указанием номеров
print('\nВведённый массив:')
for (i, el) in enumerate(a):
    print('{:2}-й элемент: {}'.format(i + 1, el))
```

Поиск максимума (минимума) в массиве

```
# a - массив чисел, len(a) > 0
a_max = a[0]
for i in a:
    if i > a_max:
        a_max = i
```

max(a)

```
# a - массив чисел, len(a) > 0
i_max = 0
for i in range(1, len(a)):
    if a[i] > a[i_max]:
        i_max = i
```

a.index(max(a))

Матрицы

В математике - таблица чисел

В программировании - массив массивов (двумерный массив)

a = [[1, 2, 3], [4, 5, 6]]

N x M: N - количество строк, M - количество столбцов

Обращение к элементу: a[i][j] # i - строка, j - столбец

Создание матриц

```
a = [[0] * m] * n # неправильно!!! создастся n ссылок на 1-ю строку
```

```
a = [[0] * m for i in range(n)] # правильно
```

Ввод-вывод матриц

```
n = int(input('Введите количество строк матрицы: '))
m = int(input('Введите количество столбцов матрицы: '))
a = []
for i in range(n):
    a.append([])
    for j in range(m):
        a[i].append(int(input('Введите {}-й элемент {}-й строки: '.format(i+1, j+1))))
```

Отладка

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки

Простейший инструмент отладки - отладочная печать (отладочный вывод).