Методы последовательностей

sequence of T – последовательность элементов типа T.

array of T, HashSet<T>, SortedSet<T> являются последовательностями.

Генерация последовательностей

… → **последовательность**

```
Range(a,b: integer): sequence of integer
Range(a,b,step: integer): sequence of integer
Range(c1,c2: char): sequence of char
Partition(a,b: real; n: integer): sequence of real
SeqRandomInteger(n: integer[; a,b: integer]): sequence of integer;
SeqRandomReal(n: integer[; a,b: real]): sequence of real;
Seq(params a: array of T): sequence of T;
SeqFill(count: integer; x: T): sequence of T;
SeqGen(count: integer; f: integer -> T): sequence of T;
SeqGen(count: integer; first: T; next: T -> T): sequence of T;
SeqGen(count: integer; first,second: T; next: (T,T) -> T): sequence of T;
SeqWhile(first: T; next: T -> T; pred: T -> boolean): sequence of T;
SeqWhile(first,second: T; next: (T,T) -> T; pred: T -> boolean): sequence of T;
SeqGen(count: integer; f: integer -> T): sequence of T;
```

То же для масивов с заменой Seq на Arr – возвращают **array of** T

Группа 1. Вывод последовательностей

```
Print(delim: string := ' '): sequence of T;
Println(delim: string := ' '): sequence of T;
```

Группа 2. Фильтрация, инвертирование

sequence of T → sequence of T

```
Where(T -> boolean)
Where((T, integer) -> boolean)
Take(count)
TakeLast(count)
Skip(count)
SkipLast(count)
TakeWhile(T -> boolean)
TakeWhile((T, integer) -> boolean)
SkipWhile(T -> boolean)
SkipWhile((T, integer) -> boolean)
Slice(from,step[,count]: integer): sequence of T;
Distinct
Reverse
```

Группа 3. Проецирование

sequence of T → sequence of TRes

```
Select(T -> TRes)
Select((T, integer) -> TRes)
SelectMany(T -> sequence of TRes)
SelectMany((T, integer) -> sequence of TRes)
```

Группа 4. Упорядочивание

sequence of T → упорядоченная sequence of T

```
Sorted
SortedDescending
OrderBy(T -> TKey)
OrderByDescending(T -> TKey)
ThenBy(T -> TKey)
ThenByDescending(T -> TKey)
```

Группа 5. Вычисление скаляра

sequence of T → скалярный тип

```
Count([T -> boolean]): integer
Average: double
Average(T -> числовой_тип): double
Sum: числовой_тип
Sum(T -> числовой_тип): числовой_тип
Max: T
Max(T -> TRes): TRes
Min: T
Min(T -> TRes): TRes
MinBy(selector: T -> TKey): T
MaxBy(selector: T -> TKey): T
Aggregate((T, T) -> T): T
Aggregate(TRes seed, (TRes, T) -> TRes): TRes
JoinIntoString([delim: string]): string;
```

Группа 6. Вычисление логического значения

sequence of T → boolean

```
All(T -> boolean)
Any(T -> boolean)
Contains(x: T) // можно x in s
SequenceEqual(second: sequence of T)
```

Группа 7. Сцепление и операции над множествами

sequence of T → sequence of T

```
Concat(second: sequence of T) // можно s1 + s2 или a1 + a2
Union(second: sequence of T)
Intersect(second: sequence of T)
Except(second: sequence of T)
```

Группа 8. Объединение, разделение

sequence of TOuter → sequence of TRes

```
Zip(second: sequence of T, (T,T)->T1): sequence of T1
Cartesian(second: sequence of T1): sequence of (T,T1);
ZipTuple(second: sequence of T1): sequence of (T,T1);
sequence of (T,T1).UnZipTuple: (sequence of T,sequence of T1);
SplitAt(ind: integer): (sequence of T,sequence of T);
Partition(cond: T->boolean): (sequence of T,sequence of T);
Interleave(second: sequence of T): sequence of T;
Numerate([from: integer]): sequence of (integer,T);
Tabulate(F: T->T1): sequence of (T,T1);
Pairwise: sequence of (T,T);
```

Группа 9. Преобразование в контейнер

sequence of T → коллекция определенного типа

```
ToArray: array of T;
ToHashSet: HashSet<T>;
ToSortedSet: SortedSet<T>;
```

Группа 10. Поэлементные операции

sequence of T → T

```
First([T -> boolean])
Last([T -> boolean])
Single([T -> boolean])
ElementAt(integer index)
```

Группа 11. Действия над элементами

```
ForEach(action: T -> ())
```