Лекция #9

Основы программирования **Последовательности**

Основы алгоритмизации и программирования

Mayer Svetlana Fyodorovna

Последовательности

Последовательность - это набор элементов, которые могут обрабатываться друг за другом

Последовательность:

- Не хранит все свои элементы в памяти
- В текущий момент в памяти есть только один элемент
- Последовательность - это алгоритм для получения значений по одному (последовательно по очереди)
- Основной цикл для работы - **foreach**

Последовательность

Последовательность имеет тип **sequence of** … (… - простой тип).

Массивы и списки – тоже своего рода последовательности.

Это значит, что:

- Массивы и списки могут быть присвоены последовательностям
- Методы последовательности подходят для массивов и списков

```
begin
  var s: sequence of integer;
  var a: array of integer := Arr(1,2,3);
  s := a;
  var l: List<integer> := new List<integer>(a);
  s := l;
  s.Println;
end.
```

Println – это метод последовательностей!

Ввод и дальнейшая обработка последовательностей

- ReadSeqInteger
- ReadSeqReal

Накапливается последовательность, а переменная q хранит ее

```
begin
var q:=ReadSeqReal(10);
var s:=0.0;
foreach var x in q do
  s+=x;
print(s)
end.
```

Принцип замены

Принцип замены. Мы можем использовать любой вид последовательности (массив или список) вместо последовательности, при этом функциональность остается такая же.

```
function SumSquares(s: sequence of integer): integer;
begin
  Result := 0;
  foreach var x in s do
    Result += x*x;
end;

begin
  SumSquares(Arr(1,2,3)).Print; // 1*1 + 2*2 + 3*3
  SumSquares(Lst(2,3,4)).Print;
end.
```

```
14 29
```

Генерация последовательностей

```
Seq(1,3,5,7,9) .Print; // 1 3 5 7 9

print(Range(1,10)); // [1 2 3 4 5 6 7 8 9 10]

print(Range(1,10,2)); // [1 3 5 7 9]
print(Range('a','k')); // [a,b,c,d,e,f,g,h,i,j,k]  sequence of char

var q:=SeqRandomInteger(5,10,20);
print(q); // [12,18,16,14,16]

var q:=SeqRandomReal(3, 1.0, 5.0);
print(q); // [4.98996168374548,2.22339218166815,2.81110574389394]

print(SeqFill(10,55)); // 55 55 55 55 55 55 55 55 55 55

print(Partition(0.0, 6.0, 4));
// divide equally into 4 parts [0, 1.5, 3, 4.5, 6]
```

Во всех случаях элементы последовательности не сохраняются в памяти. Они просчитываются по мере необходимости

Tasks

- Task 1,2

Методы последовательностей

```
s.Sum
s.Average
s.Min
s.Max
s.Count
s.Count(condition)
s.All(condition)
s.Any(condition)
s.Print
s.Println
s.PrintLines
```

Пример

Задача: Создайте функцию, которая ищет в последовательности количество максимальных элементов.

```
function findCountMax(a: sequence  of integer):integer;
begin
var k:=a.Max();
  foreach var i in a do
    if i=k then result+=1;
end;
begin
  var c:=Seq(1,5,2,10,1,10);
  c.Println();
  println('кол-во максимальных',findCountMax(c));
end.
```

```
1 5 2 10 1 10
кол–во максимальных 2
```

Пример

Задача: Определите последовательность, используя метод **Seq**. Вычислите количество встреченного в последовательности введенного числа. Использовать метод count (условие).

```
begin
  var c:=Seq(-1,2,3,-5,2,-7,8,2,11);
  c.Println();
  var x:=readinteger('введите x:');
  var n:=c.count(c->c=x);
  println($' {x} встречается в последовательности{n} раз');
end.
```

```
-1 2 3 -5 2 -7 8 2 11
введите x: 2
2 встречается в последовательности 3 раз
```

Генераторы последовательностей с лямбда выражениями

```
SeqGen(10,i->i); // 0 1 2 3 4 5 6 7 8 9

SeqGen(10,i->i,1); // 1 2 3 4 5 6 7 8 9 10
```

- SeqGen(count: integer; f: integer -> T): sequence of T;

```
begin
var sq:=SeqGen(5,x->x+1);
sq.println; // 1 2 3 4 5
end.
```

- SeqGen(count: integer; first: T; next: T -> T): sequence of T;

```
begin
var sq:=SeqGen(5, 4, x->x+1);
sq.println; // 4 5 6 7 8
end.
```

Во всех случаях элементы последовательности не сохраняются в памяти. Они просчитываются по мере необходимости

Генераторы последовательностей с лямбда выражениями

- SeqGen(count: integer; first, second: T; next: (T,T) -> T): sequence of T;

```
begin
var sq:=SeqGen(5, 1, 3, (x,y)->x+y);
sq.println; // 1 3 4 7 11
end.
```

- SeqWhile(first: T; next: T -> T; pred: T -> boolean): sequence of T;

Condition is added:

```
begin
var sq:=seqWhile(2,x->x*2,x->x<=1024);
sq.println; // 2 4 8 16 32 64 128 256 512 1024
end.
```

- SeqWhile(first,second: T; next: (T,T) -> T; pred: T -> boolean): sequence of T;

Во всех случаях элементы последовательности не сохраняются в памяти. Они просчитываются по мере необходимости

Пример

- **Задача:** Создать последовательность из N чисел Фибоначчи

```
begin
var n:=readInteger('Введите n');
var sq:=SeqGen(n,1,1,(x,y)->x+y);
sq.println();
end.
```

```
Введите n 8
1 1 2 3 5 8 13 21
```

Пример

- **Задача:** Создать последовательность из N чисел, сгенерированные в результате итеративного процесса :

$a_1=2,\ a_k=(a_{k-1}+1)*a_{k-1},\ k = 2,3,\ldots$

```
begin
var n:=readInteger('Введите n');
var sq:=SeqGen(n,2,x->(x+1)*x);
sq.println();
end.
```

```
Введите n 9
2 6 42 1806 3263442 -1461943274 -757910022 5287454 1232959906
```

Бесконечные генераторы последовательностей

- Cycle()

Повторение блока последовательности

Take is used to restrict

```
Seq(1,2,10,5).Cycle().Take(15).Println; // 1 2 10 5 1 2 10 5 1 2 10 5 1 2 10
```

- Repeat

Бесконечная последовательность чисел

```
var q:=55.Repeat.Take(10).Println; // 55 55 55 55 55 55 55 55 55 55
```

- Step

Генерация бесконечной последовательности с шагом

```
var q:=5.Step(2).Take(10).Println; // 5 7 9 11 13 15 17 19 21 23
```

- Iterate

Генерация бесконечной последовательности с лямбда-выражением

```
var q:=10.Iterate(x->x-2).Take(10).Println; // 10 8 6 4 2 0 -2 -4 -6 -8
```

Tasks

- Task 6,7,8,9,10,11,12

Q & A