Конференция «Использование системы программирования PascalABC.NET в обучении программированию» 2021

Динамические массивы в языке PascalABC.NET

Как и чему учить?

Пучкин Максим Валентинович

Старший преподаватель каф. прикладной математики и программирования ЮФУ

mpuchkin@sfedu.ru

Что такое «массив»?

- «*Упорядоченный набор однотипных элементов (данных), каждый из которых имеет свой уникальный номер (индекс). К любому элементу массива можно обратиться, указав имя массива и номер конкретного элемента*». Достаточно ли этого определения?
- «Упорядоченность» – для любой пары элементов можно указать, какой из них находится «раньше» в последовательности, а какой – «позже».
- Индексы элементов непрерывны, «без дыр», и диапазон индексов указывается при объявлении массива.
- Элементы массива располагаются в виде непрерывного блока в памяти компьютера, и обеспечивают произвольный, очень быстрый доступ к элементу.
- Входит в триаду «массивы, строки, файлы» – фундаментальные структуры в обучении программированию.

Проблемы обучения

- Массивы – как правило, первые «серьёзные» структуры данных, с которыми мы работаем. Строки и кортежи уже можем использовать в работе, однако без подробностей.
- Не с чем сравнивать, да и вообще целесообразность сравнения структур данных под вопросом. Нет (и долго не будет) задач, которые затрагивают вопросы эффективности, производительности, и прочего.
- В PascalABC.NET нет адресной арифметики – она не нужна.
- Многие особенности работы могли бы быть объяснены с учётом расположения элементов в памяти. Традиционно массив – последовательно расположенные в памяти элементы одинакового размера. Однако в PascalABC.NET это не работает, и в действительности совершенно непонятно, как и что там в памяти расположено – детали реализации скрыты.
- Выбиваются из стилистики коллекций – List<T>, HashSet<T> и прочих. При этом довольно большая часть таких коллекций доступна в PascalABC.NET.

Индексация массивов

- Классический Borland Pascal допускал использование произвольного диапазона при индексации – статические массивы. PascalABC.NET для поддержки совместимости работает с такими массивами. При этом такие массивы практически не используются, однако литературы и материалов довольно много.
- Учим только zero-based массивам с индексацией, остальные не нужны!

```
const Size = 20;
type Mas = array [1..Size] of integer;
var A : Mas;
    s : integer;

begin
  for var i := 1 to Size do
    Read(A[i]);
  for var i := 1 to Size do
    s := s + A[i];
  Writeln('Сумма элементов массива :', s);
end.
```

```
const Size = 20;
var A : array of integer;
    s : integer;
begin
  SetLength(A, Size);

  for var i := 0 to Size-1 do
    Read(A[i]);
  for var i := 0 to Size-1 do
    s := s + A[i];
  Writeln('Сумма элементов массива :', s);
end.
```

Создание массивов

- Для знакомства с массивами предлагается использовать несколько простых способов создания массивов:

```
var A := |1,2,3,0,10,5|;
Println(A[2]);

var B := Arr(1,3,5,7);
Println(B[1]);

var C := Arr(1..10);
Println(C[2]);

var D : array of integer := (0,1,2,3);
Println(D[3]);

var E : array of integer := |1,2,3|;
Println(E[0]+E[1]);
```

Первые три способа оптимальны для первых шагов

Последние два способа явно неудачны и избыточны для начинающих!

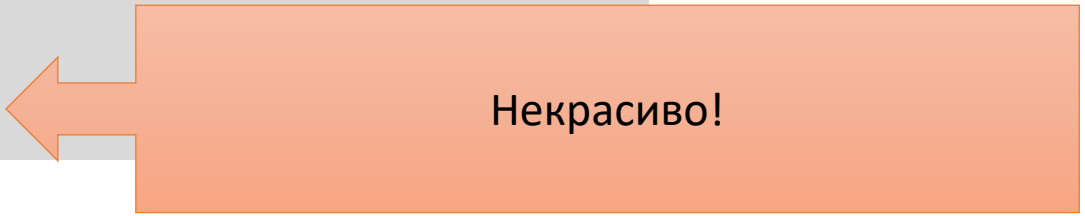
Свойства Length и High

- Динамические массивы «знают» свой размер, а также максимальный индекс элементов:

```
var A := new integer[10];
Println('Длина массива равна', A.Length);
Println('Индекс последнего элемента равен', A.High);

Println('Введите', A.Length, 'элементов массива :');
for var i := 0 to A.High do
  Read(A[i]);

for var i := 0 to A.High do
  Print(A[i]);
```



- К сожалению, диссонанс с коллекциями, использующими свойство Count, но до этого ещё далеко.

Операции с массивами

- «*С массивами целиком нельзя делать ничего, кроме присваивания! Можно присвоить один массив другому, всё остальное – поэлементная работа!*» (© один молодой преподаватель).

```
begin
  var A := |1,2,3|;
  var B := Arr(10..20);
  var C := A + B;
  Println(C);

  A := A*3 + |10,11,12|*3;
  Println(A);

  A := ArrRandomInteger(10,2,5);
  Println(A);

  A := ReadArrInteger(10);
  Println(A);
end.
```

Сумма массивов

Сумма и произведение массивов

Генерация массива случайных чисел

Чтение массива из 10 элементов

Алгоритмы

- Массивы – ссылочный тип данных, который можно передавать в процедуры и функции «по значению» (без **var**):

```
//  Циклический сдвиг влево
procedure ShiftLeft(A : array of integer);
begin
  var tmp := A[0];
  for var i := 1 to A.High do
    A[i-1] := A[i];
  A[A.High] := tmp;
end;

begin
  var A := Arr(10..20);
  Println(A);
  ShiftLeft(A);
  Println(A);
end.
```

Массив передаётся «по значению», изменяется, и всё прекрасно работает

Алгоритмы - 2

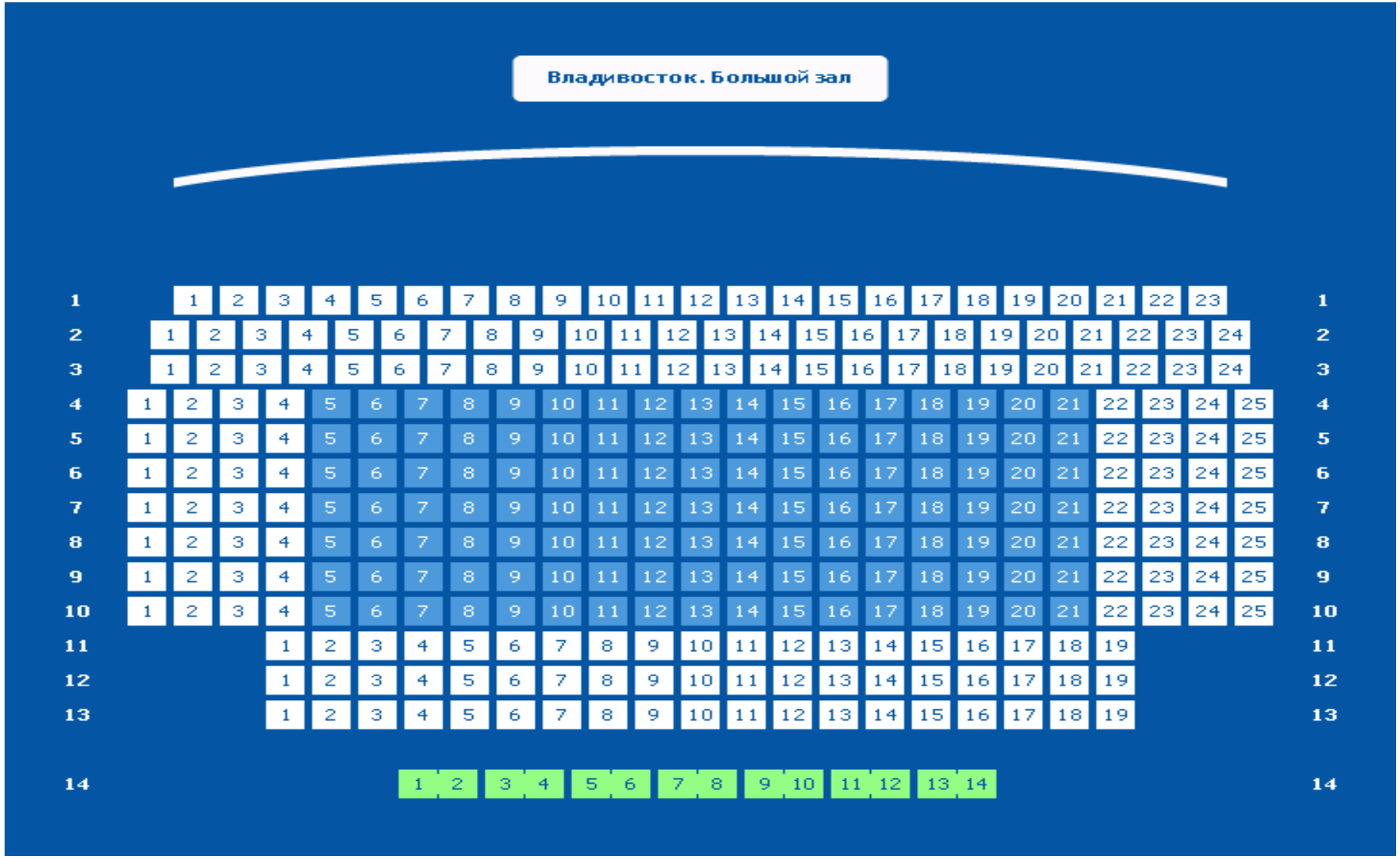
- «Массивы – ссылочный тип данных…», но не всё так просто:

```
//  Удаление чётных чисел из массива
procedure RemoveEven(var A : array of integer);
begin
  var dest := 0;
  for var source := 0 to A.High do
    if A[source].IsOdd then
    begin
      A[dest] := A[source];
      dest += 1;
    end;
  SetLength(A, dest);
end;

begin
  var A := Arr(10..20);
  Println(A);
  RemoveEven(A);
  Println(A);
end.
```

При изменении размера массива происходит перемещение в памяти!

Как учить?



Изображение с сайта https://kino.vl.ru/theatres/vladivostok/

Копирование массивов

- Семантика некоторых операций неочевидна для обучающихся. Например, сумма массивов даёт новый массив, а присваивание – нет.

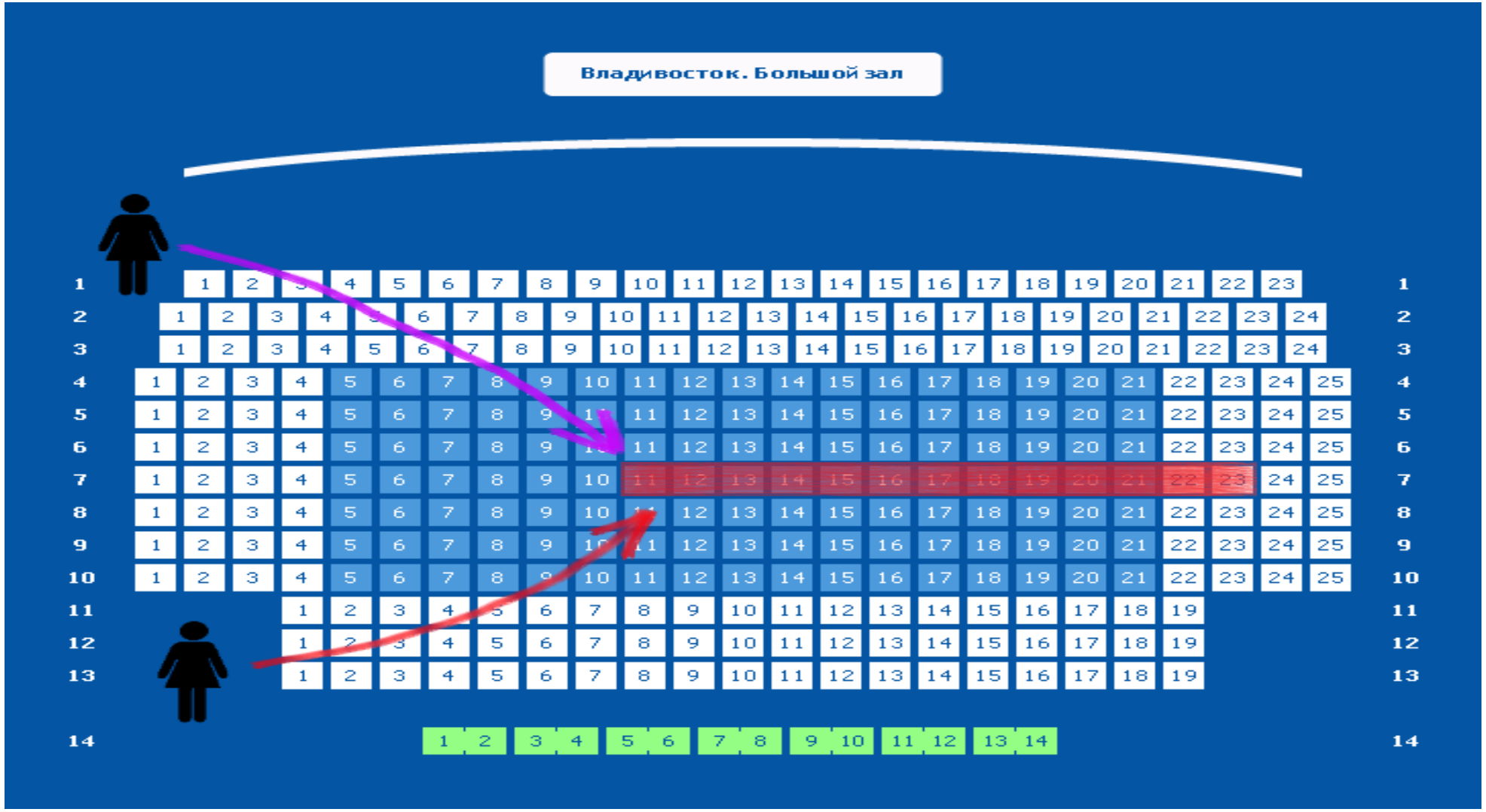
```
begin
  var A := Arr(10..20);
  Println(A);
  //  Это присваивание доставит нам хлопот
  var B := A;
  B[2] := 777;
  Println(A);

  //  Абсолютно законная операция
  B := A + A;
  B[3] := 10001;
  Println(A);

  //  Копирование массивов выполняется таким образом:
  var C := Arr(A);
end.
```

Присваивание одного массива другому приводит просто к копированию ссылки на существующий массив, копия не создаётся!

Почему нельзя копировать массивы?



Изображение с сайта https://kino.vl.ru/theatres/vladivostok/

Сравнение массивов на равенство

- Сравнивать массивы напрямую нельзя – это просто сравнение ссылок, и не работает никогда.

```
begin
  var a := Arr(1,2,1);
  var b := Arr(a[::-1]);

  Println(a);
  Println(b);

  Println(a=b);

  //  Сравнение массивов на равенство
  if a.ArrEqual(b) then
    Println('Массив - палиндром!')
  else
    Println('Массив не палиндром!')
end.
```

Для сравнения массивов используем специальный метод, проверяющий значения поэлементо. Для строк такие проблемы не актуальны.

Вставка и удаление элементов

- Для обычных массивов – «тяжелые» операции, и лучше бы их избегать, равно как и изменения размера массивов.
- В действительности алгоритмов, требующих вставки или удаления элементов, не так уж много, и они довольно специфичны. Более того, традиционно под эти операции оптимизированы другие структуры данных, которые выходят далеко за границы школьной программы.

```
//  Удаление чётных чисел из массива
procedure RemoveAt(var A : array of integer; index : integer);
begin
  for var i := index+1 to A.High do
    A[i-1] := A[i];
  SetLength(A, A.Length-1);
end;
```

Методы массивов

- Довольно мощный набор стандартных методов, уже реализованы для массивов. Основная проблема – реализовывать эти методы самостоятельно, а потом использовать, либо же сразу использовать стандартные.

a.Length – длина массива

a.Min – минимальный элемент в массиве

a.Max – максимальный элемент в массиве

a.IndexMin – индекс первого минимального элемента в массиве

a.IndexMax – индекс первого максимального элемента в массиве

a.Sum – сумма элементов в числовом массиве

a.Product – произведение элементов в числовом массиве

a.Average – среднее элементов в числовом массиве

a.First – первый элемент в массиве

a.Last – последний элемент в массиве

a.IndexOf(x) – индекс первого значения x или -1 если не найдено

a.Replace(x,y) – заменить в массиве все значения x на y

Срезы

- Срезы имеют вид:

a[from:to]

a[from:to:step]

a?[from:to]

a?[from:to:step]

Представляют из себя «подмассив» – фрагмент исходного массива, возможно, с некоторым шагом.

```
begin
  var A := Arr(1..10);
  Println(A[2:4]);
  Println(A[0:A.Length:2]);
  A := A[:A.Length div 2];  //  Половина массива
  A.Println;
  A := A[::-1];  //  Инверсия массива
  A.Println;
end.
```

Вставка и удаление элементов срезами

- Использование срезов для модификации массивов тоже является не самым лучшим решением. Однако это настолько удобная и простая концепция, что школьники вполне нормально её воспринимают, и можно сконцентрироваться на интересных задачах.
- Для задач ЕГЭ любые вопросы производительности и эффективности роли не играют, т.к. Python всё «закрывает».

```
begin
  var A := Arr(1..10);
  Println(A);
  //  Вставка элемента в позицию 5
  A := A[:5] + |777| + A?[5:];
  Println(A);

  //  Удаление элемента из позиции 6
  A := A[:6] + A?[7:];
  Println(A);
end.
```

Итоги

- Массивы – самая объёмная тема в обучении программированию, при использовании срезов и запросов. И самая интересная!
- В PascalABC.NET можно выстроить любую траекторию обучения, с совершенно разной глубиной и уровнем погружения в тему.
- Рекомендуется использование только динамических массивов, статические – не тема для школьников. Однако эта рекомендация не абсолютна – большой объём литературы по Borland Pascal и строки всё портят.
- Адресная арифметика и способы хранения массивов в памяти остаются «за бортом», однако в виде примеров эти темы можно затрагивать.
- Вопросы эффективности и быстродействия не возникают у школьников практически никогда, и на это можно не оглядываться. Однако плохому учить тоже не лучшая идея.

Спасибо за внимание!