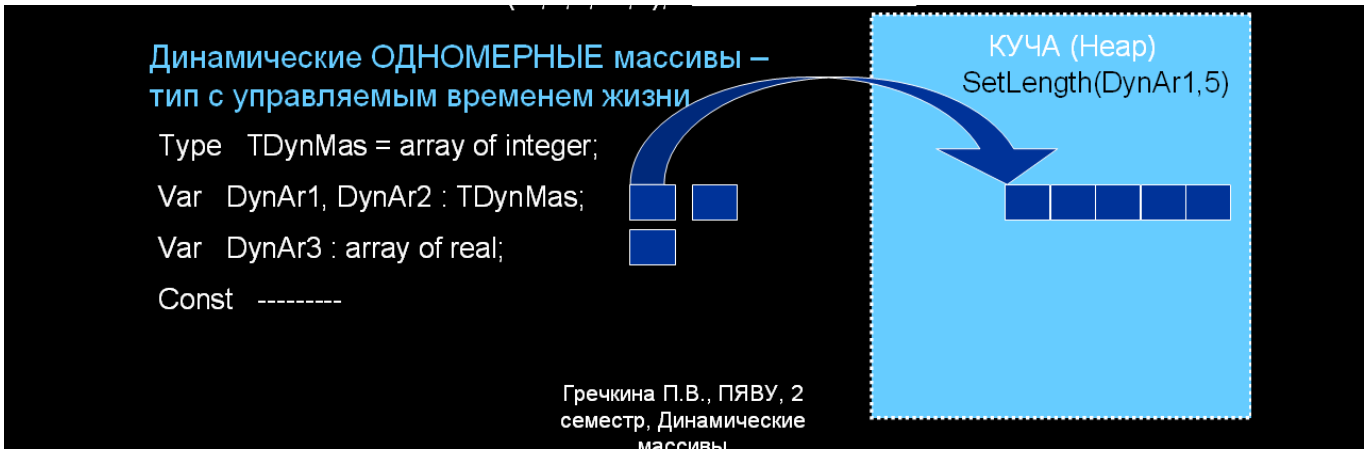
Лекция 4. Динамические массивы в Delphi

В языка Delphi существуют тип открытого динамического массива, при объявлении типа которого, не указывается правая граница (остается открытой), но зато известна левая - это всегда 0.

Нельзя объявить константу динамического типа.

Динамический открытый массив является *типом с управляемым временем жизни*, как и длинные строки. И память для него не выделяется и не освобождается *напрямую*, как со своими типами указателей (см. Лекцию 3), а требует только «намека» в программе, чтобы при трансляции и компиляции был добавлен нужный код выделения памяти и освобождения памяти по *технологии «сборки мусора»* при выходе из области видимости или при обнулении указателя массива.

Для выделения памяти надо указать желаемый размер(ы) с помощью процедуры SetLength:



При присваивании одному динамическому массиву другого происходит копирование адреса. Количества ссылок на массив (как у строк) нет, и массивы будут указывать на одну память, если сам программист не укажет явно – создать копию – с помощью функции Copy.

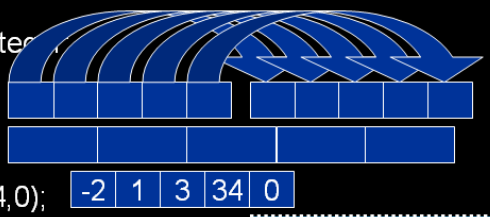
А вот при присваивании одного *статического* массива другому произойдет копирование:

Динамические массивы

Статические ОДНОМЕРНЫЕ массивы

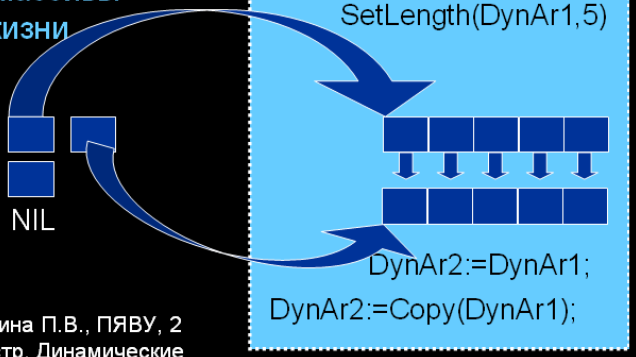
```
Type   TStatMas = array ['a'..'e'] of inte
Var    StatAr1, StatAr2 : TStatMas;
Var    StatAr3 : array [1..5] of real;
Const StatAr4: TStatMas = (-2,1,3,34,0);
```

StatAr2:=StatAr1;



Динамические ОДНОМЕРНЫЕ массивы – тип с управляемым временем жизни

```
Type   TDynMas = array of integer;
Var    DynAr1, DynAr2 : TDynMas;
Var    DynAr3 : array of real;
Const  ---------
```



Перед вводом значений элементов массива надо **обязательно выделить память для них! И не забыть про индекс с 0**

Динамические массивы

Ввод массива

```
Uses SysUtils;
var N, i:byte;
    A: array of integer;
    Sum: integer;  s: string;
begin
  write('N=?'); readln(N);
  SetLength(A,N);

  For i:=0 to N-1 do
    read(A[i]);
  readln;

  Sum:=0;
  For i:=Low(A) to High(A) do
       Sum:=Sum+A[i];

  writeln(Format('Ответ: %d',[Sum])); //SysUtils %3d %5.2f %s
end.
```

Динамические массивы

Ввод массива

```
Uses SysUtils;
var N, i:byte;
    A: array of integer;
    Sum: integer;  s: string;
begin
  write('N=?'); readln(N);
  SetLength(A,N);

  For i:=0 to N-1 do
    read(A[i]);
  readln;

  Sum:=0;
  For i:=Low(A) to High(A) do
       Sum:=Sum+A[i];

  writeln(Format('Ответ: %d',[Sum])); //SysUtils %3d %5.2f %s
end.
```

```
For i:=0 to N-1 do  // с обнулением некорректных:
Begin
  readLN(s);
  if not TryStrToInt(s, A[i]) then A[i]:=0; // SysUtils
      // TryStrToFloat ( string, extended ) c ,  → 3,14
end;
```

Гречкина П.В., ПЯВУ, 2 семестр, Динамические массивы 4

При использовании открытого массива в качестве параметра процедур есть свои особенности.

Для одномерного открытого массива не обязательно описывать свой тип, но от этого зависит способ передачи параметра, описанного с ключевым словом **out.,** и возможность передать **статический** массив вместо **динамического.**

Динамические массивы

Параметр процедуры – массив констант

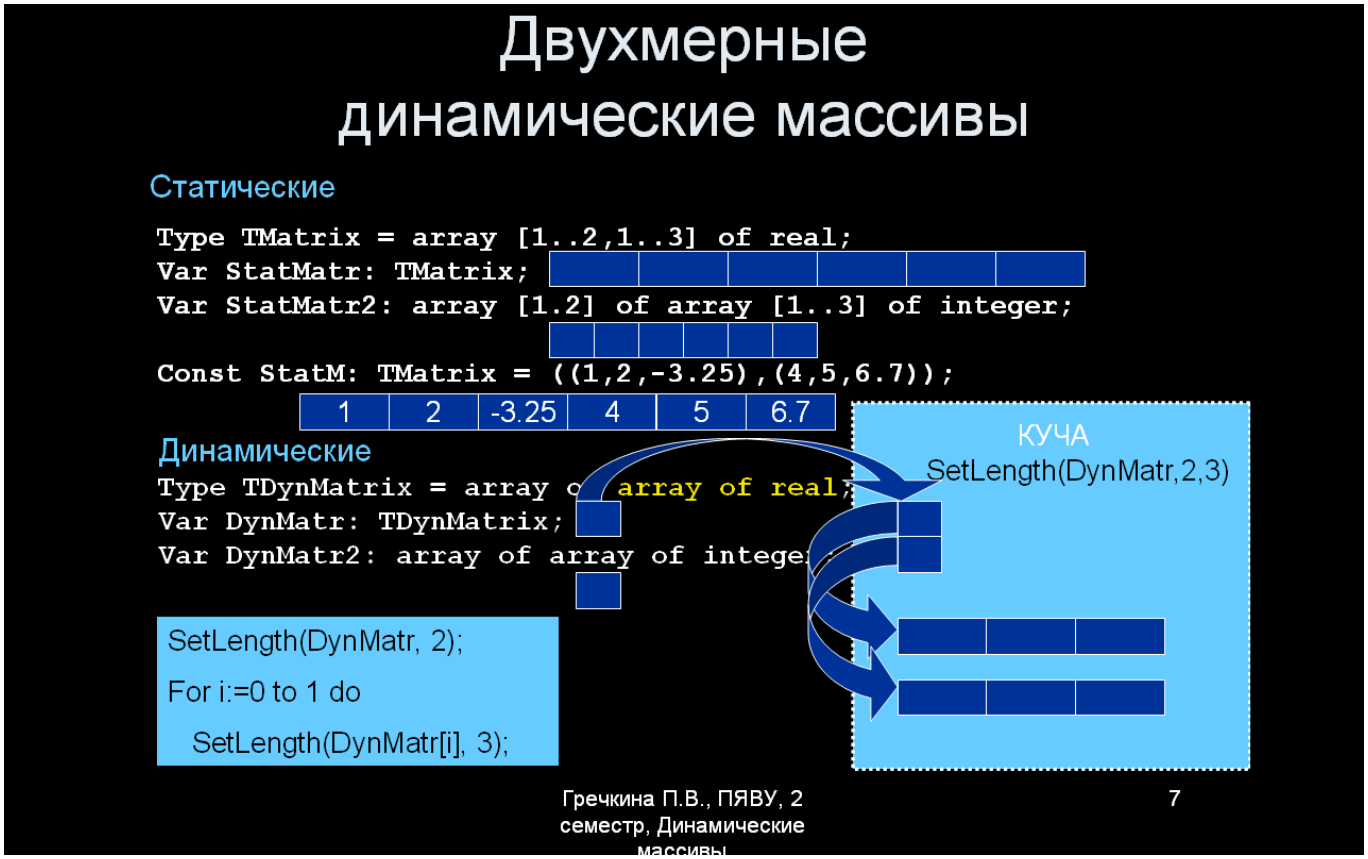
```
function kolvo_otr3(const x: array of const): integer;
var i,kol: integer;
begin
  kol:=0;
  for i:=0 to High(x) do
    if (x[i].VType = vtInteger) then
      if x[i].VInteger < 0 then inc(kol);
  kolvo_otr3:=kol;
end;
```

```
K:= kolvo_otr(DynAr1);

K:= kolvo_otr2(DynAr1);
K:= kolvo_otr2(StatAr4);
K:= kolvo_otr2([2,-3,4,5,6]);

K:= kolvo_otr3([2,'stroka','c',-4,-3.5]);
```

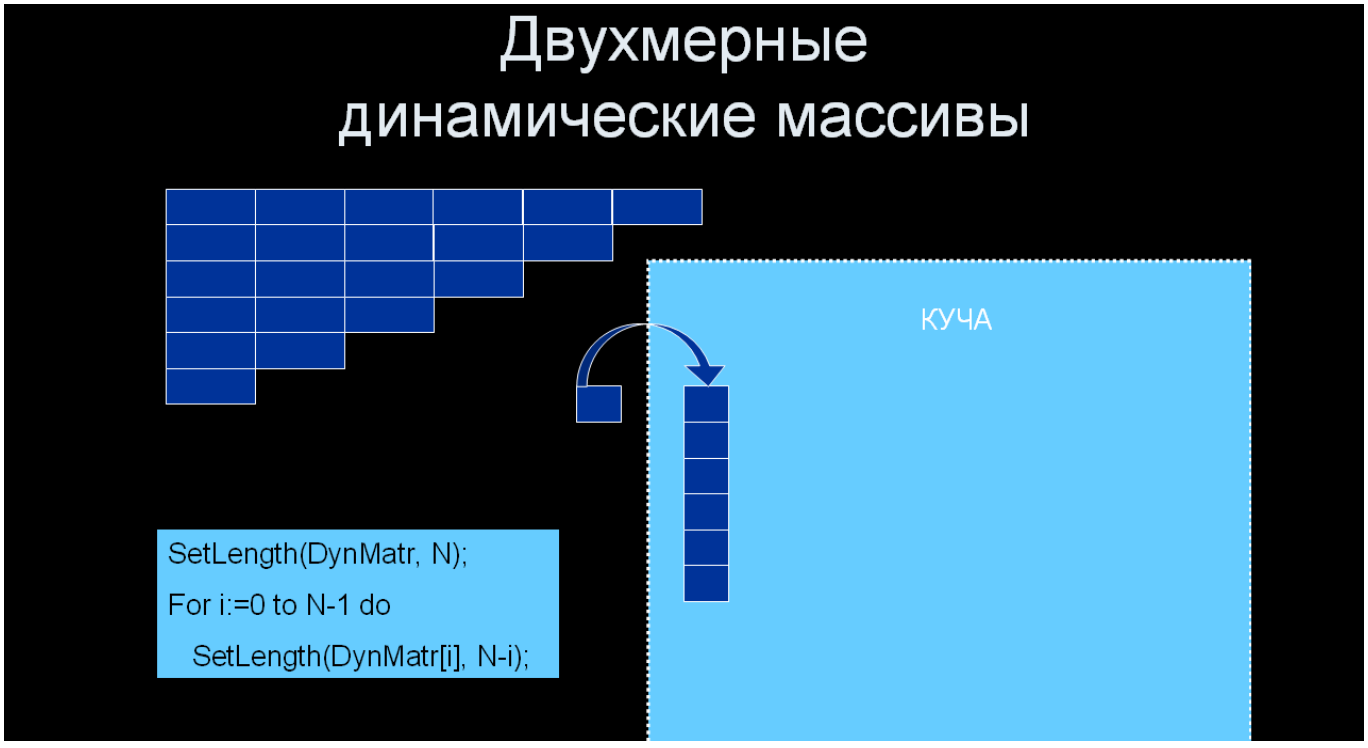
В квадратных скобках здесь указано не множество, а конструктор массива.

Массив из констант – это фактически массив ив из элементов типа **Variant.**

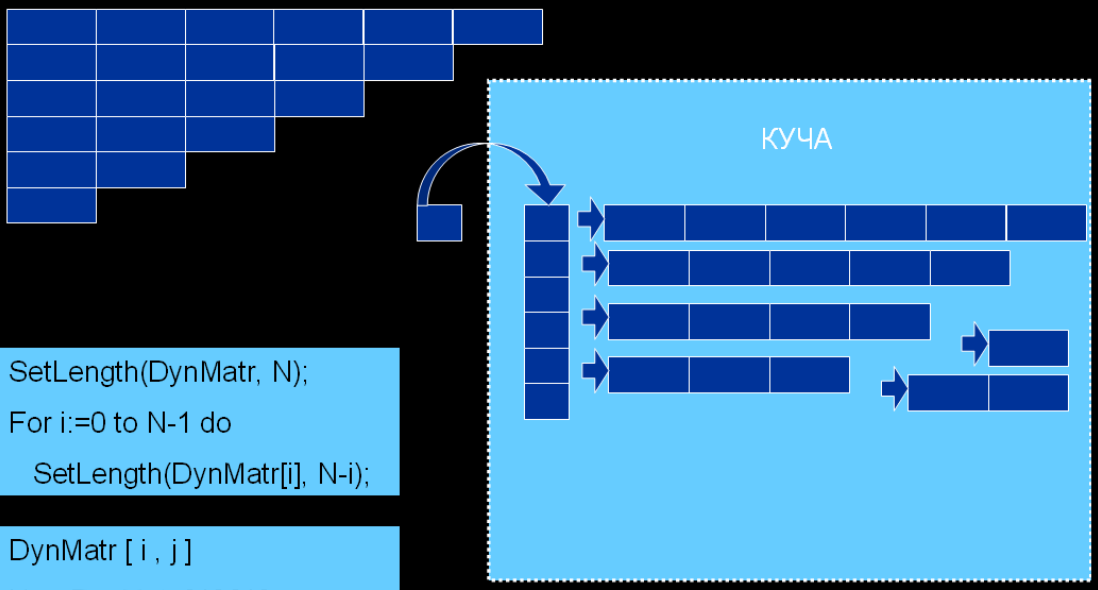
Статические двухмерные и динамические двухмерные массивы размещаются в памяти по-разному, и НЕ совместимы при передаче параметров:



Такое отдельное расположение строк открытого двухмерного массива в памяти позволяет создавать Непрямоугольные матрицы, а , например, треугольную:



Двухмерные динамические массивы




```
SetLength(DynMatr, N);
For i:=0 to N-1 do
  SetLength(DynMatr[i], N-i);
```

```
DynMatr [ i , j ]
Или DynMatr [ i ] [ j ]
```


Обращение к элементам статического и динамического массива записывается одинаково (хоть и двумя способами, подходящими для любого вида массива).

Передача строк и массивов в качестве параметра-константы:

Динамические массивы и строки как параметры процедур

Type TStr = ANSIString; TDynMas = array of real;

	String	TStr	Array of real	TDynMas
const	(вх) Совместима с короткими строками	(вх) Совместима с короткими строками	(вх) Совместим со статич. одномерн. массивом	Значения меняются! но не его размер/адрес, он – чисто вх

```
Procedure p11(const s:string); =>  p11(ANSIs); p11(ShortS);  p11('Stroka');
Procedure p12(const s:Tstr); =>  p12(ANSIs); p12(ShortS);  p12('Stroka');
Procedure p13(const a: array of real); =>  p13(DynA); p13(StatA);  p13([2, -4, -1.5]);
Procedure p14(const a: TDynMas); =>  p14(DynA);
```

Передача строк и массивов в качестве параметра-значения:

Динамические массивы и строки как параметры процедур

Type TStr = ANSIString;

	String	TStr	Array of real	TDynMas
-- (по умолчанию)	(вх+пром) Совместима с короткими строками	(вх+пром) Совместима с короткими строками	(вх+пром) Совместим со статич. одномерн. массивом (станет с 0 индекс)	Значения меняются! но не его размер/адрес (тогда созд-ся локал. копия)

Procedure p21(s:string); => p21(ANSIs); p21(ShortS); p21('Stroka');

Procedure p22(s:Tstr); => p22(ANSIs); p22(ShortS); p22('Stroka');

Procedure p23(a: array of real); => p23(DynA); p23(StatA); p23([2, -4, -1.5]);

Procedure p24(a: TDynMas); => p24(DynA);

Передача строк и массивов в качестве параметра-переменной:

Динамические массивы и строки как параметры процедур

Type TStr = ANSIString; TDynMas = array of real;

	String	TStr	Array of real	TDynMas
var	(вх+пром+вых)	(вх+пром+вых)	(вх+пром+вых)	(вх+пром+вых)
out	=/= var (пром+вых) Память освобождает-ся при входе!	=/= var (пром+вых) Память освобождает-ся при входе!	= var	=/= var (пром+вых) Память освобождает-ся при входе!

Procedure p41(out s:string); => p41(ANSIs);

Procedure p42(out s:Tstr); => p42(ANSIs);

Procedure p43(out a: array of real); => p43(DynA); p43(StatA);

Procedure p44(out a: TDynMas); => p44(DynA);

При использование ключевого слова out освобождается память для динамических типов с управляемым временем жизни. Освобождения нет в третьем случае, так как

таким способом может быть передан и статический массив, освободить память, выделенную для которого невозможно в процессе выполнения программа, так как при статическом распределении памяти она выделяется один раз, и не из Кучи.

Примеры, демонстрирующие работу с разными динамическими массивами:

```
program PrjDynMas;

{$APPTYPE CONSOLE}

uses
  SysUtils;

Type // одномерные динамические массивы
  DynMas1 = array of real; // первый способ – открытый массив
  Mas = array [1..1] of real;
  DynMas2 = ^Mas; // второй способ (индексы с 1) – свой тип
  DynMas3 = ^Real; // третий способ как в Си – свой тип массива

var
  dm1: DynMas1;
  dm2: Dynmas2;
  dm3, tek: DynMas3;
  i: byte;
begin
  try
     randomize;

    {--------------------dm1--------------------------}
    SetLength(dm1, 5);
    for i:=0 to 4 do dm1[i]:=1+i + random(10)/10;

    write('dm1 :');
    for i:=0 to 4 do write(dm1[i]:3:1, ' ');
    writeln;

    //dm1:=nil;

    {--------------------dm2--------------------------}
    GetMem(dm2, 5*SizeOf(real));
    for i:=1 to 5 do dm2^[i]:=i + random(10)/10;

    write('dm2 :');
    for i:=1 to 5 do write(dm2^[i]:3:1, ' ');
    writeln;

    FreeMem(dm2, 5*SizeOf(real));

    {--------------------dm3--------------------------}
    GetMem(dm3, 5*SizeOf(real));
    tek:=dm3;
    for i:=0 to 4 do
    begin
      Dynmas3(Cardinal(dm3)+i*SizeOf(real))^:=1+i + random(10)/10;
     // tek^:=1+i + random(10)/10;
      tek:=Dynmas3(Cardinal(dm3)+i*SizeOf(real));
      //tek:=Dynmas3(Cardinal(tek)+SizeOf(real));
```

```
    end;

    write('dm3 :');
    for i:=0 to 4 do
      write(Dynmas3(Cardinal(dm3)+i*SizeOf(real))^:3:1, ' ');
    writeln;

    FreeMem(dm3, 5*SizeOf(real));

  except
    on E:Exception do
      Writeln(E.Classname, ': ', E.Message);
  end;
  readln
end.
```

```
dm1 :1.9 2.9 3.0 4.8 5.3
dm2 :1.4 2.8 3.5 4.8 5.8
dm3 :1.9 2.9 3.4 4.1 5.2
```

```
program PrjDynMas2;
{$APPTYPE CONSOLE}

uses
  SysUtils;

type
  DynMas1 = array of array of real;
  ///Mas = array [1..1,1..1] of real;   не сработает
  ///DynMas2 = ^Mas;
  DynMas3 = ^Real;

var
  dm1, dm2: DynMas1;
  dm3, tek: DynMas3;
  i,j: byte;
begin
  try
     randomize;

    {--------------------dm1---------------------------}
    SetLength(dm1, 3, 3);
    for i:=0 to 2 do
      for j:=0 to 2 do dm1[i,j]:=1+j +i*3 + random(10)/10;

    writeln('dm1 :');
    for i:=0 to 2 do
    begin
      for j:=0 to 2 do write(dm1[i][j]:3:1, ' ');
      writeln;
    end;

    //dm1:=nil;

    {--------------------dm2---------------------------}
    SetLength(dm2, 3);
    for i:=0 to 2 do  SetLength(dm2[i], i+1); //с разной длиной строк

    for i:=0 to 2 do
      for j:=0 to High(dm2[i]) do dm2[i,j]:=1+j +i*3 + random(10)/10;
```

```
    writeln('dm2 :');
    for i:=0 to 2 do
    begin
      for j:=0 to High(dm2[i]) do write(dm2[i][j]:3:1, ' ');
      writeln;
    end;

    //for i:=0 to 2 do dm2[i]:=nil;
    //dm2:=nil;

    {--------------------dm3--------------------------}
    GetMem(dm3, 3*3*SizeOf(real));
    for i:=0 to 2 do
    for j:=0 to 2 do
    begin // вычисление адреса элемента по индексам от начала массива
      Dynmas3(Cardinal(dm3)+(i*3+j)*SizeOf(real))^:=
1+j+3*i + random(10)/10;
    end;

    writeln('dm3 :');
    for i:=0 to 2 do
    begin // вычисление адреса элемента по индексам от начала массива
      for j:=0 to 2 do
        write(Dynmas3(Cardinal(dm3)+(i*3+j)*SizeOf(real))^:3:1, ' ');
      writeln;
    end;

    writeln('dm3 :');
    tek:=dm3; // вычисление адреса элемента через текущий элемент
    for i:=0 to 2 do
    begin
      for j:=0 to 2 do
      begin
        write(tek^:3:1, ' ');
        inc(tek); // или  tek:=Dynmas3(Cardinal(tek)+SizeOf(real));
      end;
      writeln;
    end;

    FreeMem(dm3, 3*3*SizeOf(real));

  except
    on E:Exception do
      Writeln(E.Classname, ': ', E.Message);
  end;
  readln
end.
```

```
dm1 :
1.4 2.8 3.1
4.2 5.4 6.6
7.7 8.0 9.4
dm2 :
1.6
4.9 5.4
7.1 8.3 9.2
dm3 :
1.5 2.0 3.5
4.6 5.2 6.0
7.9 8.0 9.3
dm3 :
1.5 2.0 3.5
4.6 5.2 6.0
7.9 8.0 9.3
```