Динамические структуры данных

Указатели

Статические данные

```
var x, y: integer;
    z: real;
    A: array[1..10] of real;
    str: string;
```

- переменная (массив) имеет **имя**, по которому к ней можно обращаться
- **размер** заранее известен (задается при написании программы)
- память выделяется **при объявлении**
- размер **нельзя увеличить** во время работы программы

Динамические данные

- размер заранее неизвестен, определяется во время работы программы
- память выделяется во время работы программы
- нет имени?

Проблема:

как обращаться к данным, если нет имени?

Решение:

использовать адрес в памяти

Следующая проблема:

в каких переменных могут храниться адреса?
как работать с адресами?

Указатели

Указатель – это переменная, в которую можно записывать адрес другой переменной (или блока памяти).

Объявление:

указатель

```
var pC: ^char;     // адрес символа
    pI: ^integer; // адрес целой переменной
    pR: ^real;     // адрес вещ. переменной
```

Как записать адрес:

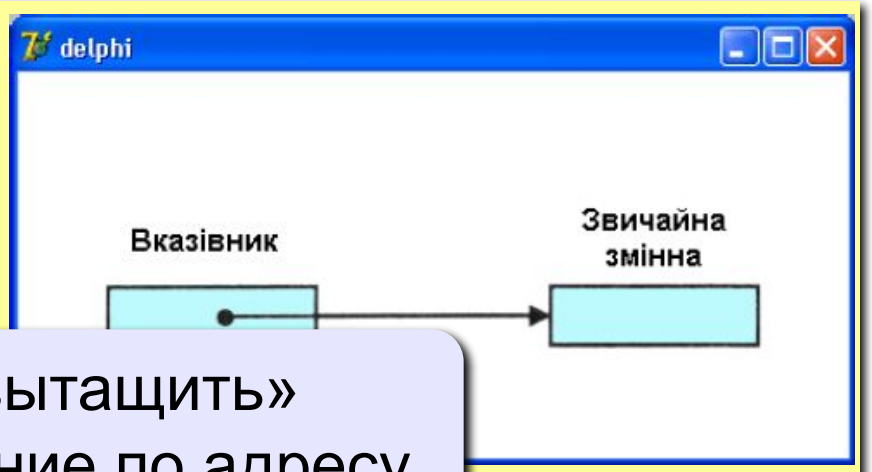
```
var m: integer;     // целая переменная
    pI: ^integer;  // указатель
  A: ar[1..10] of integer; // массив
...
pI:= @m;     // адрес переменной m
pI:= @A[1]; // адрес элемента массива A[1]
pI:= nil;    // нулевой адрес
```

адрес ячейки

1

Обращение к данным через указатель

```
...
var m, n: integer;
    pI: ^integer;
begin
  m := 4;
  pI := @m;
  writeln('Адрес m = ', pI); // вывод адреса
  writeln('m = ', pI^); // вывод значения
  n := 4*(7 - pI^);     // n = 4*(7 - 4) = 12
  pI^ := 4*(n - m);     // m = 4*(12 - 4) = 32
end.
```



«вытащить» значение по адресу

Обращение к данным (массивы)

```
var i: integer;
    A: array[1..4] of integer;
    pI: ^integer;
...
begin
  for i:=1 to 4 do A[i] := i;
  pI := @A[1]; // адрес A[1]
  while ( pI^ <= 4 ) // while( A[i] <= 4 )
    do begin
    pI^ := pI^ * 2;  // A[i] := A[i]*2;
    pI := pI + 1; // к следующему элементу
  end;
end.
```

Не работает в *PascalABC.NET*!

переместиться к следующему элементу = изменить адрес на **sizeof(integer)**

Что надо знать об указателях

- указатель – это переменная, в которой можно хранить адрес другой переменной;
- при объявлении указателя надо указать тип переменных, на которых он будет указывать, а перед типом поставить знак `^`;
- знак `@` перед именем переменной обозначает ее адрес;
- запись `p^` обозначает *значение* ячейки, на которую указывает указатель **p**;
- `nil` – это *нулевой указатель*, он никуда не указывает
- при изменении значения указателя на **n** он в самом деле сдвигается к **n**-ому следующему числу данного типа (для указателей на целые числа – на `n*sizeof(integer)` байт).

Нельзя использовать указатель, который указывает неизвестно куда (будет сбой или зависание)!

Динамические структуры данных

Динамические массивы

Где нужны динамические массивы?

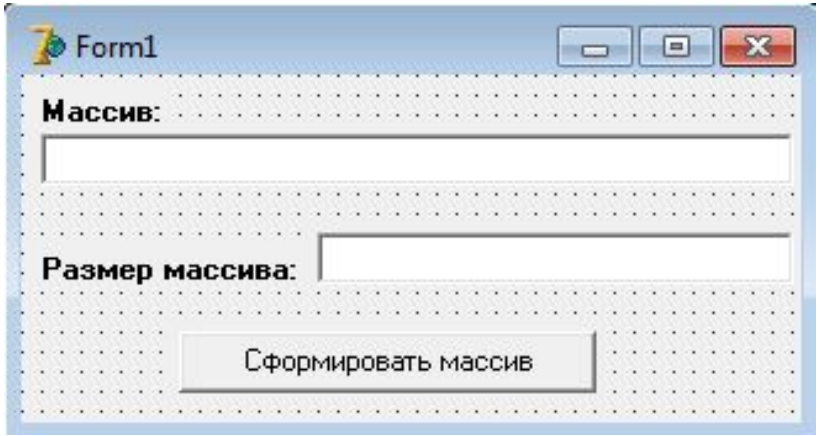
Задача. Ввести размер массива, затем – элементы массива. Отсортировать массив и вывести на экран.

Проблема:

размер массива заранее неизвестен.

Пути решения:

1) выделить память «с запасом»;
2) выделять память тогда, когда размер стал известен.



Алгоритм:

3) ввести размер массива;
4) **выделить память**
5) ввести элементы массива;
6) отсортировать и вывести на экран;
7) **удалить массив**

Динамические массивы *(Delphi)*

```
procedure TForm1.Button1Click(Sender:
TObject);
var A: array of integer;
    i, n: integer;
begin
  n:=StrToInt(Edit2.Text);
  SetLength ( A, n );
  for i := 0 to N-1 do
    A[i]:=random(20);
  ... { сортировка }
  for i := 0 to N-1 do
    Edit1.Text:=Edit1.Text + IntToStr(A[i])+'
';
  SetLength( A, 0 );
end.
```

какой-то массив целых чисел

выделить память

нумерация с **НУЛЯ**!

освободить память

Динамические массивы *(Delphi)*

- при объявлении массива указывают только его тип, память не выделяется:

```
var A: array of integer;
```

- для выделения памяти используют процедуру **SetLength** *(установить длину)*

```
SetLength ( массив, размер );
```

- номера элементов начинаются с **НУЛЯ**!
- для освобождения блока памяти нужно установить нулевую длину через процедуру **SetLength**:

```
SetLength ( массив, 0 );
```

Ошибки при работе с памятью

Запись в «чужую» область памяти:

память не была выделена, а массив используется.

Что делать: так не делать.

Выход за границы массива:

обращение к элементу массива с неправильным номером, при записи портятся данные в «чужой» памяти.

Что делать: если позволяет транслятор, включать проверку выхода за границы массива.

Указатель удаляется второй раз:

структура памяти нарушена, может быть все, что угодно.

Что делать : в удаленный указатель лучше записывать `nil`, ошибка выявится быстрее.

Утечка памяти:

ненужная память не освобождается.

Что делать : убирайте «мусор»

(в среде .NET есть сборщик мусора!)

Динамические матрицы *(Delphi)*

Задача. Ввести размеры матрицы и выделить для нее место в памяти во время работы программы.

Проблема:

размеры матрицы заранее неизвестны

Решение:

```
...
var A: array of array of integer;
    N, M: integer;
begin
  N:=StrToInt(Edit1.Text);// число строк
  M:=StrToInt(Edit2.Text);// число столбцов
  SetLength ( A, N, M );
  ... // работаем, как с обычной матрицей
  SetLength( A, 0, 0 );
end.
```

Динамические структуры данных

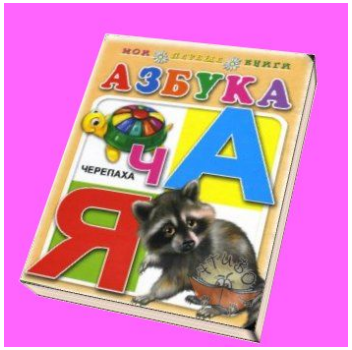
Динамические переменные

Динамические структуры (записи)

Динамические переменные

```
procedure TForm1.Button1Click(Sender: TObject);
var
p1,p2,p3:^integer; //указатели на переменную integer
begin
  // создаем динамические переменные типа integer
  // выделяем память для динамических переменных
  New(p1);  // выделение памяти под переменную p1
  New(p2);
  New(p3);
  p1^ := 5;
  p2^ := 3;
  p3^ := p1^ + p2^;
  ShowMessage('Сумма чисел равна' + IntToStr(p3^));
  // уничтожаем динамические переменные
  // выделяем память, занимаемую этими переменными
  Dispose(p1);
  Dispose(p2);
  Dispose(p3);
end;
```

Структуры (в Паскале – *записи*)



Свойства:

- автор (*строка*)
- название (*строка*)
- год издания (*целое число*)
- количество страниц (*целое число*)

Задача: объединить эти данные в единое целое

Структура (запись) – это тип данных, который может включать в себя несколько *полей* – элементов разных типов (в том числе и другие структуры).

Размещение в памяти

автор	название	год издания	количество страниц
40 символов	80 символов	целое	целое

Новый тип данных – запись

Объявление типа:

Память не выделяется!

```
type TBook = record
    author: string[40]; // автор, строка
    title:  string[80]; // название, строка
    year: integer;// год издания, целое
    pages : integer;    // кол-во страниц, целое
  end;
```

TBook – *Type Book («тип книга»)* – удобно!

Статическое объявление переменных и массивов:

```
const N = 10;
var Book: TBook; // одна запись
    aBooks: array[1..N] of TBook;  // массив
```

Динамическое выделение памяти под запись

```
var pB: ^TBook;
begin
  New(pB);
  pB^.author := 'А.С. Пушкин';
  pB^.title := 'Полтава';
  pB^.year := 1990;
  pB^.pages := 129;
  Dispose(pB);
end.
```

переменная-указатель на **TBook**

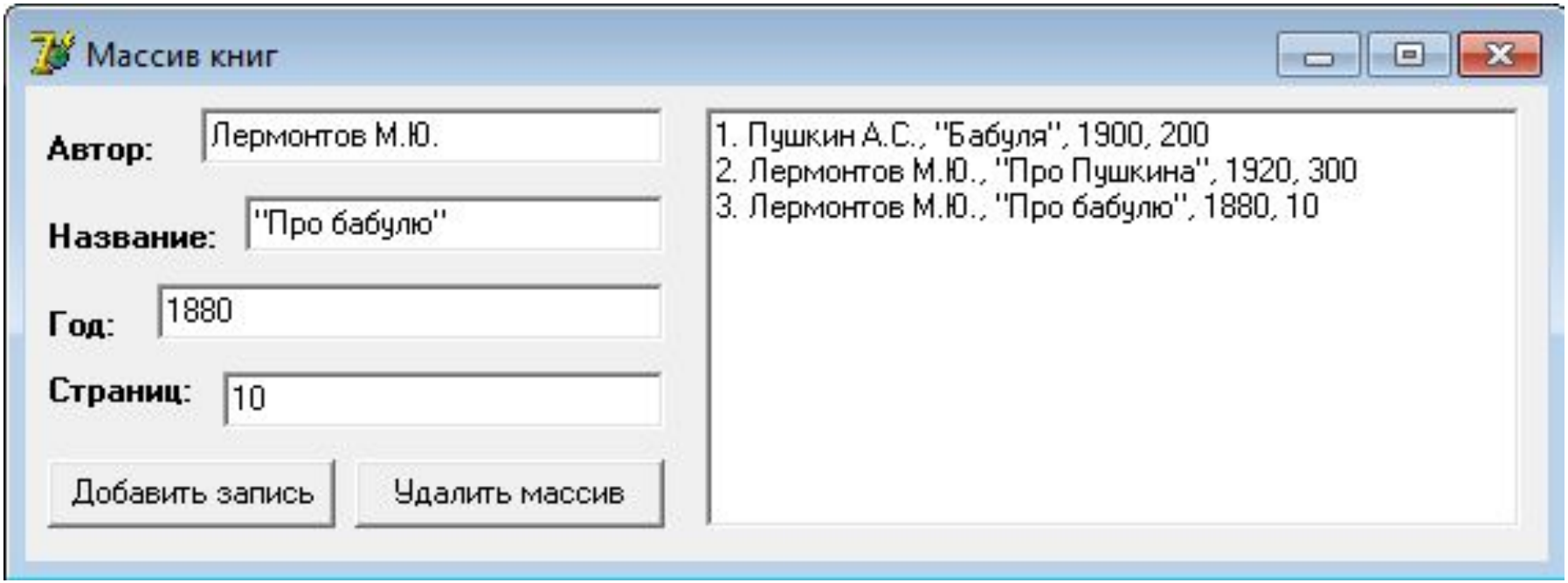
выделить память под запись, записать адрес в **pB**

Для обращения к полю записи по адресу используется знак ^

освободить память

Динамическое выделение памяти под элементы массива

Задача. **Создать массив книг. Предусмотреть ввод каждой книги с клавиатуры. Память под каждый новый элемент массива должна выделяться динамически.**
Предусмотреть вывод массива, а также функции освобождения памяти от занимаемого массива.



Реализация в программе

```
type
  PBook = ^TBook; { новый тип данных }
  TBook = record
    author: string[40]; // автор, строка
    title:  string[80]; // название, строка
    year: integer;// год издания, целое
    pages : integer;    // кол-во страниц, целое
  end;
var
  Form1: TForm1;
  p: array[1..100] of PBook; {массив указателей на тип
TBook}
  kol:integer; {текущее количество элементов в массиве}
...
{процедура создания формы (при запуске программы)}
procedure TForm1.FormCreate(Sender: TObject);
begin
  kol:=0;{обнуляем начальное количество элементов
массива}
```

Реализация в программе

```
{процедура нажатия кнопки «Добавить запись»}
procedure TForm1.Button1Click(Sender: TObject);
var
  i:integer;
begin
  kol:=kol+1; {количество элементов массива +1}
  new(p[kol]);{создание элемента массива, адрес в
p[kol]}
  p[kol]^.author:=Edit1.Text; //заполнение элемента
  p[kol]^.title:=Edit2.Text;   // массива
  p[kol]^.year:=StrToInt(Edit3.Text);  // данными
  p[kol]^.pages:=StrToInt(Edit4.Text);  // из формы
  Memo1.Clear;  // очистка Memo1
  for i:=1 to kol do  //вывод массива в Memo
    Memo1.Lines.Add(IntToStr(i)+'. '+p[i]^.author+', `
    +p[i]^.title+', '+IntToStr(p[i]^.year)+', `
    +IntToStr(p[i]^.pages));
end;
```

Реализация в программе

```
{процедура нажатия кнопки «Удалить массив»}
procedure TForm1.Button2Click(Sender: TObject);
var
  i:integer;
begin
  for i:=1 to kol do
    Dispose(p[i]); {удаление i-го элемента массива}
  Memo1.Clear;
  kol:=0; {обнуление кол-ва элементов массива}
end;
```