

10. Вычисление по формулам (продолжение)

10.6. Уточнение многоместных (n - арных) операций

$$S = U_1 \circledast U_2 \circledast U_3 \circledast \dots \circledast U_n$$

обобщенный вид
n - арной операции

$$S = \sum_{i=1}^n x_i = x_1 + x_2 + x_3 + \dots + x_n$$

$$S = \prod_{i=1}^n x_i = x_1 \cdot x_2 \cdot x_3 \cdot \dots \cdot x_n$$

$$S = X^N = \underbrace{x \cdot x \cdot x \cdot \dots \cdot x}_{n \text{ раз}}$$

$$S = X! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \dots \cdot X$$

\circledast - так обозначается обобщенная бинарная (двухместная) операция ("+", "-", ".", "возведение в степень"). Исполнитель алгоритма, на которого ориентирована программа, к сожалению многоместные операции выполнять не может (имеет только одно вычислительное устройство) и поэтому вычисление многоместных операций нужно заменять на последовательность бинарных операций.

Для решения поставленной задачи составим соответствующий линейный алгоритм (заменим одну многоместную операцию набором 2-местных операций):

$S_1 = U_1$
 $S_2 = S_1 \circledast U_2$
 $S_3 = S_2 \circledast U_3$
 $S_4 = S_3 \circledast U_4$
 \vdots
 $S_n = S_{n-1} \circledast U_n$

Можно заметить, что строки, начиная со второй, можно записать в следующей обобщенной форме (на Псевдокоде):
 для i от 2 до n повторить
 $S_i = S_{i-1} \circledast U_i$ - рекуррентная формула

В этой последовательности начиная со второй строки хорошо видна однотипность вычислений. Поэтому, заметив эту однотипность, попытаемся найти обобщенную запись. Обобщенная запись последовательности шагов, на каждом из которых последующее значение результата получается с использованием предыдущего, называется рекуррентной формулой (от слова рекурсия). Рекурсия в общем случае - это выражение (определение) какого-то понятия самого через себя. В данном случае осуществляется вычисление нового значения переменной S через ее старое значение (i -е промежуточное значение S вычисляется через $(i-1)$ -ое). Если удалось получить такую обобщенную формулу, то это означает, что можно свернуть группу действий (для которой удалось найти обобщенную запись) в цикл (с заранее известным числом повторений). В общем случае цикл будет состоять из трех частей (порядок их важен):

| |
|------------|
| Подготовка |
| Заголовок |
| Тело |

В тело цикла включается то, что удалось записать в виде рекуррентной формулы, т.е. тело цикла - это те действия, которые надо многократно повторять. Заголовок цикла задает закон изменения т.н. параметра цикла, из которого определяется количество повторения цикла (из приведенного выше заголовка цикла видно, что цикл будет выполняться $n-1$ раз для i от 2 до n). В подготовку цикла включаются те действия, которые не удалось подвести под рекуррентную формулу. Подготовка цикла настраивает цикл на первое правильное выполнение. На Псевдокоде цикл для выполнения рассмотренной выше последовательности действий имеет вид:

подготовка — $S_1 = U_1$
 заголовок — Для i от 2 до n повторить
 тело цикла — $S_i = S_{i-1} \circledast U_i$

Согласно «Правил разработки циклов» (чуть позже рассмотрим) мы имеем дело с циклом с известным числом повторений, которые разрабатываются в данном случае методом «снизу вверх».

Согласно Правил после получения записи цикла необходимо попытаться выполнить упрощение подготовки цикла. Подготовка бывает простой, если в правой части оператора присваивания используется константа или переменная без индекса.

В данном случае подготовка простой не является, потому что в правой части оператора присваивания используется переменная U_1 - с индексом. Для упрощения подготовки составляется система уравнений:

- в 1-е уравнение переписывается сам оператор присваивания из подготовки цикла;
- 2-е уравнение получается из рекуррентной (обобщенной) формулы путем подстановки в нее (в левую и правую часть оператора присваивания) нужных значений (тех, которые используются в подготовке) индексов (в данном случае $i=1$):

$$\begin{array}{l} S_1 = U_1 \\ S_1 = S_0 \odot U_1 \end{array} \longrightarrow \boxed{U_1 = S_0 \odot U_1}$$

Частные случаи:

- если обобщенная операция "+", то $S_0=0$;
- если обобщенная операция "*", то $S_0=1$.

В общем случае присваивание для простой подготовки цикла должно иметь вид: $S_0=\text{const}$.

Если попытка упрощения подготовки цикла удалась, то необходимо скорректировать заголовок цикла таким образом, чтобы число повторения цикла увеличилось на единицу. Это нужно сделать потому что в результате упрощения подготовки цикла действия которые ранее выполнялись при подготовке (т.е. вне цикла), теперь включаются в тело цикла и нужно еще один проход тела цикла выполнить, что бы эти действия выполнялись.

В данном случае цикл примет следующий вид:

$S_0=\text{const}$;

Для i от 1 до n повторить

$$S_i := S_{i-1} \odot U_i$$

Далее согласно теории (См п. «Использование индексов в математике и программах») необходимо исключить лишние индексы. В результате получится следующий вид цикла.

$S = \text{const}$

Для i от 1 до n повторить

$$S = S \odot U_i$$

2

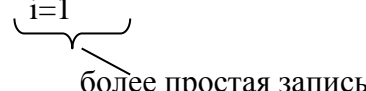
Часто бывают такие ситуации, когда компоненты многоместных операций представляют из себя не простые выражения (как в рассмотренном выше случае), а достаточно сложные.

Например:

$$S = \sum_{i=1}^n x^i$$

В данном случае нужно привести сложную запись каждого операнда многоместной операции к более простой форме, которая была до этого.

$$S = \sum_{i=1}^n x^i = \sum_{i=1}^n u_i, \text{ где } u_i = x^i$$


 более простая запись

Далее в дополнение к поиску обобщенной формулы для самой многоместной операции необходимо попытаться найти обобщенную запись для вычисления каждой сложной компоненты многоместной операции, например, в виде рекуррентной формулы вида: $U_i = f(U_{i-1})$. Самый простой традиционный способ для нахождения нужной формулы – это деление типа

$$Z = U_i / U_{i-1}.$$

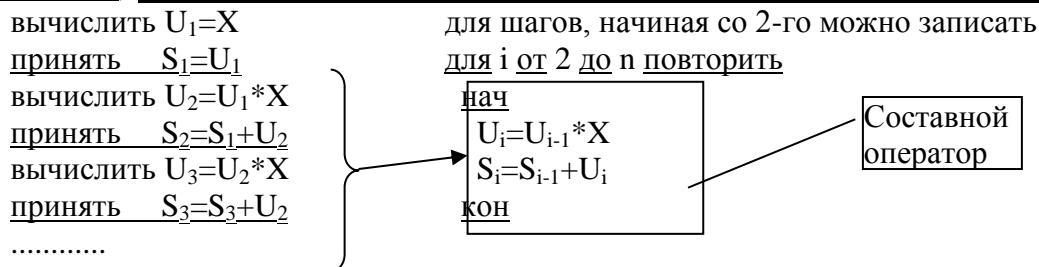
Тогда при известном Z можно вычислить U_i через U_{i-1} :

$$U_i = Z * U_{i-1}$$

Для данного примера такая формула имеет следующий вид: $U_i = U_{i-1} \cdot x$

Замечание: вместо деления аналогичным образом можно выполнить вычитание.

Если попытка нахождения обобщенной записи каждой сложной компоненты многоместной операции удалась, то дальше можно выполнять те же действия, что и для более простого предыдущего случая за одним только исключением: При записи линейного алгоритма (и при записи тела цикла) **вместо одного действия у нас на каждом этапе (на каждой итерации) будет два:**

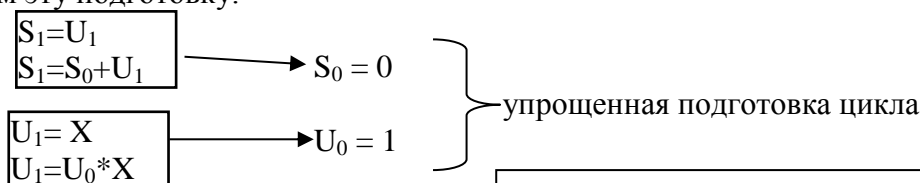


В подготовке теперь тоже будет не одно действие, а два:

1) Вычислить $U_1=X$

2) Принять $S_1 = U_1$

Упростим эту подготовку:



В итоге получим следующий цикл.

$S_0=0; U_0=1$

Для i от 1 до n повторить

нач
 $U_i=U_{i-1}*X$
 $S_i=S_{i-1}+U_i$
кон

в общем случае здесь
может быть группа
операторов или цикл

После исключения лишних индексов
получим:

$S=0; U=1$

Для i от 1 до n повторить

нач
 $U=U*X$
 $S=S+U$
кон

10.7 Подведение последовательности одинаковых действий под n -арную операцию (нахождение компактной записи).

Часто встречаются такие случаи, когда необходимо длинную последовательность действий (развернутой записи многоместной операции) подвести под цикл (под обобщенную запись):

$$S=1+3/4+4/6+5/8+\dots=1+\sum_{i=1}^n \frac{i+2}{(i+1)*2}$$

Подходов к решению этой задачи много. Самыми простейшими из них являются следующие:

а) **поделить** все следующие элементы последовательности на предыдущие и попробовать получить **геометрическую прогрессию**

Например:

$$S = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots = S = \sum_{i=1}^n a_i$$

Этот путь отличается от предыдущего тем, что вычисления на каждом шаге более простые (менее сложные) - например, умножение или возведение в степень заменяются на более простые операции - сложение

Возможны два варианта нахождения решения (нахождения вида a_i): 1) решение вида $a_i = f(i)$, т.е. нахождение формулы для прямого вычисления элементов ряда 2) решение вида $a_i = f(a_{i-1})$, т.е. нахождение формулы для вычисления текущего значения элемента ряда из предыдущего значения.

Для нахождения обоих видов решения надо составить таблицу следующего вида:

| | | | | |
|-------|---------------------------------|---------------------------------|---------------------------------|----------------------------------|
| a_i | $\frac{1}{2} = (\frac{1}{2})^1$ | $\frac{1}{4} = (\frac{1}{2})^2$ | $\frac{1}{8} = (\frac{1}{2})^3$ | $\frac{1}{16} = (\frac{1}{2})^4$ |
| i | 1 | 2 | 3 | 4 |

Из таблицы видно, что обобщенные формулы для двух видов решения имеют вид:

1) $S = 1 + \underbrace{\sum_{i=1}^n \left(\frac{1}{2}\right)^i}_{a_i}$ и 2) $S = 1 + \sum_{i=1}^n a_i$, где $a_i = a_{i-1} * (1/2)$

обобщенная запись вида $a_i = f(i) = (1/2)^i$
(здесь подготовка цикла не нужна для **вычисления** a_i - нет нужды в накапливании (сохранении ранее вычисленных значений), зато надо возводить в степень типа $\text{power}(1/2, i)$, причем результат - вещественный (хранятся приближенными); для **вычисления** S подготовка цикла нужна)

обобщенная запись вида $a_i = f(a_{i-1})$
(здесь нужна подготовка цикла для a и S :
 $S_0 := 1$;
 $a_0 := 1$)

б) **вычесть** из последующих значений ряда предыдущие и попробовать получить **арифметическую прогрессию**, т.е. представить все последующие элементы как предыдущий плюс некоторую разность

Например:

$$S = 1 + 3 + 5 + 7 + 9 + 11 = \sum_{i=1}^6 u_i$$

Для нахождения обобщенной формулы, как уже говорилось выше, **есть 2 пути**:

1 путь: угадать обобщенную формулу для прямого (без рекурсии) вычисления каждого элемента последовательности

| | | | | | | |
|-------|---|---|---|---|---|----|
| u_i | 1 | 3 | 5 | 7 | 9 | 11 |
| i | 1 | 2 | 3 | 4 | 5 | 6 |

Из таблицы видно, что $u_i = 2*i - 1$ (здесь подготовка цикла для вычисления u_i не нужна, но нужна для S - нужно $S := 0$)

В итоге получим цикл:

```
S := 0;
For i:=1 to 6 do
  S := S + 2*i - 1
```

2 путь: угадать рекуррентную формулу для вычисления u_i вида: $u_i = f(u_{i-1})$

| | | | | | | |
|-------|---|---|---|---|---|----|
| u_i | 1 | 3 | 5 | 7 | 9 | 11 |
| i | 1 | 2 | 3 | 4 | 5 | 6 |

Из таблицы видно, что $u_i = u_{i-1} + 2$ (для i от 2 до 6)

(здесь подготовка цикла нужна и для S ($S := 0$ как и раньше)

и для u_i :

$u_1 = 1$;
 $u_1 = u_0 + 2$; } упрощенная подготовка
 $u_0 = 1 - 2 = -1$

В итоге получим цикл:

```
S := 0; u := -1;
for i:=1 to 6 do
begin
  u := u + 2;
  S := S + u;
end;
```

в) попытаться подвести под цикл не всю последовательность, а ее часть, например, выкинув (исключив) из анализа первый элемент (мысленно)

Например:

$$s = 1+3+5+7+9+11 = \sum_{i=1}^6 u_i = 1 + \sum_{i=1}^5 u_i$$

Выше (в пункте б) мы рассмотрели ситуацию, когда эта последовательность подводилась под цикл с первого элемента. Теперь рассмотрим ситуацию, когда ту же последовательность будем подводить под цикл, начиная со второго элемента (первый уйдет в подготовку цикла):

| | | | | | |
|-------|---|---|---|---|----|
| u_i | 3 | 5 | 7 | 9 | 11 |
| i | 1 | 2 | 3 | 4 | 5 |

Из таблицы видно, что при прямом вычислении $u_i = 2i+1$ (для i от 1 до 5)

Здесь подготовка цикла нужна только для S ($s_0 = 1$) и не нужна для u_i .

В итоге получим цикл:

$S := 1;$

For $i:=1$ to 5 do

$S := S + 2i+1;$

Из той же таблицы видно, что при рекурсивном вычислении u_i имеем:

$u_i = u_{i-1} + 2$ (для i от 2 до 5)

Здесь подготовка цикла нужна и для S ($s_0 = 1$) и для u :

$$\left. \begin{array}{l} u_1 = 3; \\ u_1 = u_0 + 2; \end{array} \right\} u_0 = 3 - 2 = 1;$$

В итоге получим цикл (с уже исключенными лишними индексами):

$S := 1; u := 1;$

for $i:=1$ to 5 do

begin

$u := u + 2;$

$S := S + u;$

end;

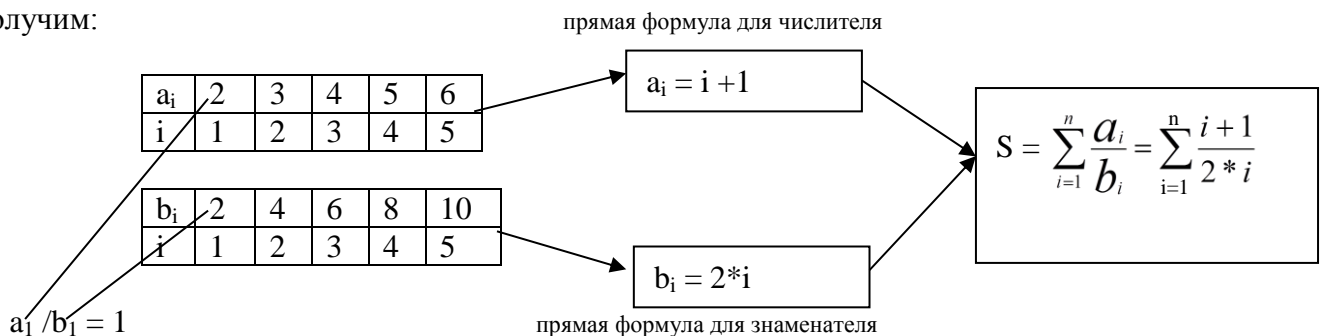
г) Если элементы последовательности - дроби и не удастся найти обобщенную запись для дробного элемента в целом, надо представить дробь в виде следующей общей записи:

$$S = \sum_{i=1}^n a_i / b_i$$

После этого отдельно для числителя и знаменателя расписывается последовательность значений. Так для случая

$$S = 1 + 3/4 + 4/6 + 5/8 + 6/10$$

получим:



Замечание:

Часто ряд значений бывает знакопеременным, например, для рассмотренного выше случая

$$S = 1 - 3/4 + 4/6 - 5/8 + 6/10 = \sum_{i=1}^5 \frac{i+1}{2*i} \underbrace{(-1)^{i+1}}_a$$

В этом случае в программе категорически не надо возводить (-1) в степень, т.е. не надо вызывать power(-1, i). Вместо этого надо завести дополнительную переменную a и делать так:

a := -1; {Знак a специально подбирается исходя из того, перед каким элементом ряда – четным или нечетным стоит знак минус}

Получим цикл:

{Подготовка цикла}

a := -1;

s := 0;

{Сам цикл}

for i = 1 to 5 do //это заголовок цикла

begin

| | |
|---------------------------|--|
| a := a * (-1); | {инвертируем - переключаем знак a при каждом повторении тела} |
| S := S + ((i+1)/(2*i))*a; | {меняем (переворачиваем) знак у слагаемого путем умножения на a} |

end;

это тело цикла

Вывод графика функции $y = x^2$ на экран в текстовом режиме:

Вариант без gotoxy()

uses

crt;

Var

y, // координата точки по оси Y (по вертикали вниз)

z, // параметр цикла for (номер точки на графике)

x : word; // координата точки по оси X (по горизонтали слева направо)

begin

clrscr;

writeln;

for z := 1 to 5 do

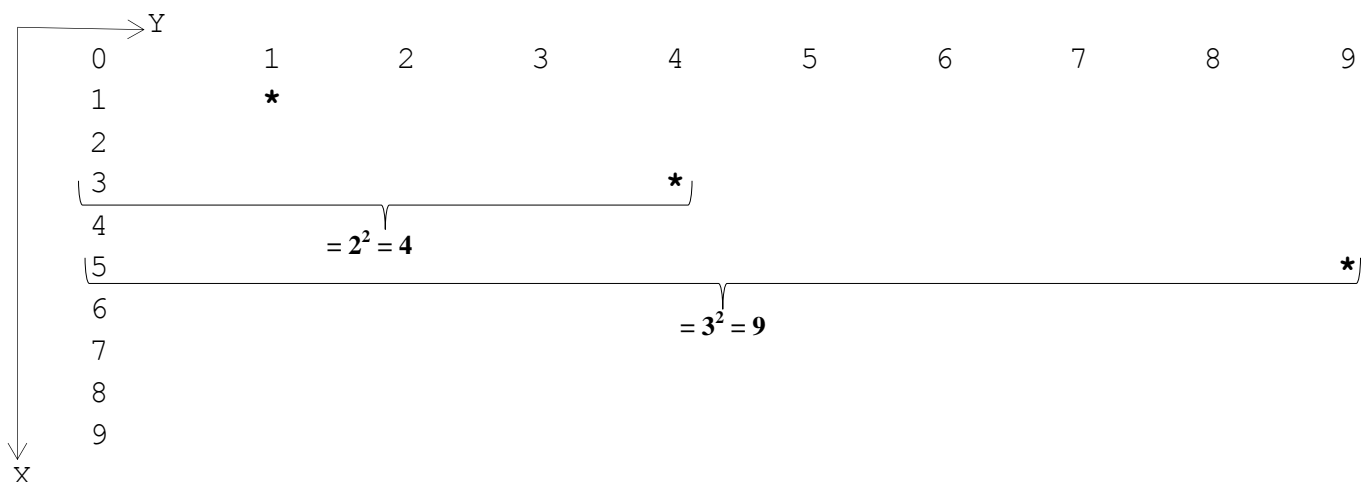
begin

writeln('*':z*z);

writeln;

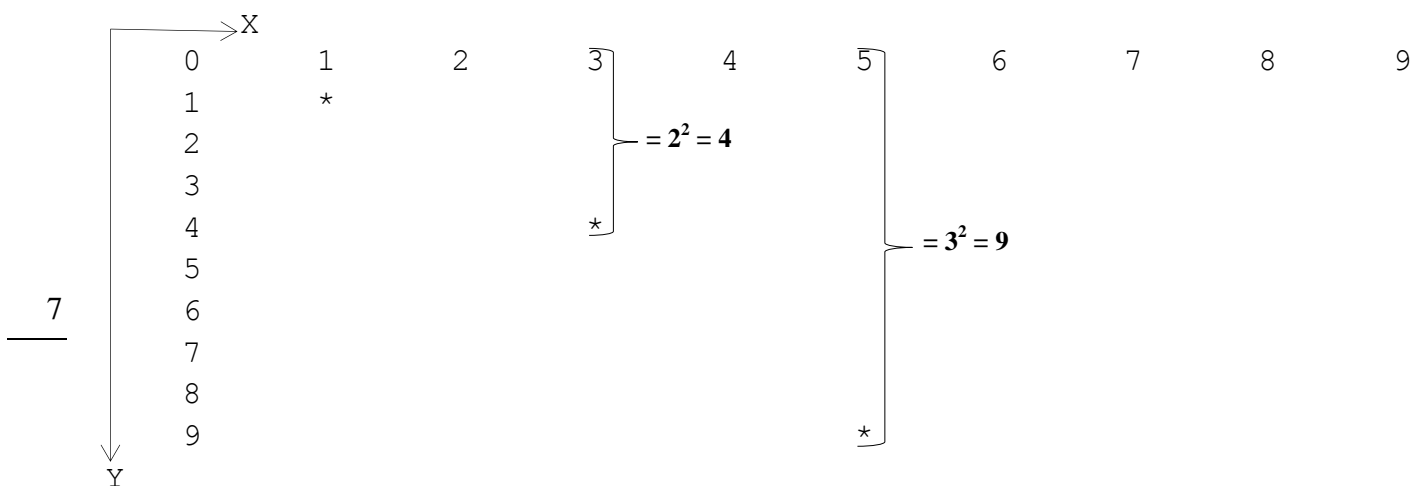
end;

end.

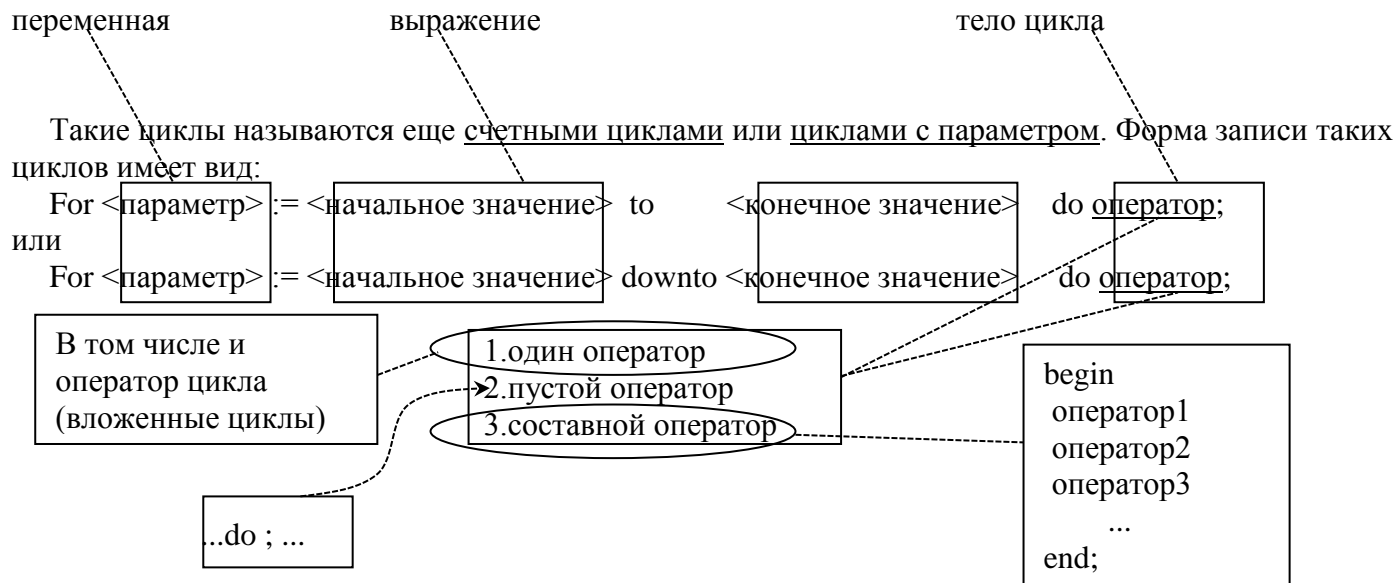


Вариант с gotoxy()

```
uses
  crt;
Var
  y,      // координата точки по оси Y (по вертикали вниз)
  z,      // параметр цикла for (номер точки на графике)
  x : word; // координата точки по оси X (по горизонтали слева направо)
begin
  clrscr;
  for z := 1 to 5 do
    begin
      x := 2 * z - 1; // x - будем проходить значения 1, 3, 5, 7, 9
      y := z * z;     // y - будем проходить значения 12, 22, 32, 42, 52
      gotoxy(x, y);
      write('*');
    end;
  end.
```



11. Средства языка Паскаль для циклов с известным числом повторений.



Параметр цикла это переменная, закон изменения которой задает количество повторений цикла. Значения параметра цикла будут изменяться от начального значения до конечного (или от конечного до начального). Шаг изменения параметра цикла фиксирован и равен единице (для верхнего примера +1, для нижнего (-1)). Параметр цикла по правилам Паскаля может быть любым порядковым типом (целым, булевым, символьным, перечисляемым, типом - диапазоном). Тело цикла будет повторяться для каждого значения параметра цикла. По выходу из цикла значение параметра цикла не определено.

Замечание: если нужно, чтобы вещественная переменная (а не порядковая) изменялась с некоторым (постоянным) шагом, то это можно сделать так:

```
Var
  r:real;
  i:integer;
begin
  r:=0.1; //начальное значение
  for i:=1 to 10 do
    begin
      r := r + 0.1;
    end;
  или
  r := (i-1) * 0.1;
  Использование r
end;
```

число повторений = ((<кон. знач.> - <нач. знач.>)/шаг) + 1

r будет изменяться от 0 до 1.0 с шагом 0.1

Логическое выражение

= 1

Порядок выполнения оператора цикла следующий:

1 шаг. Проверяется значение логического выражения:

для верхнего (to) "начальное значение" <= "конечное значение"
для нижнего (downto) "начальное значение" >= "конечное значение"

Если это логическое выражение имеет значение "истина", то параметр цикла получает начальное значение и выполняется 2 шаг (п.2.1).

Если значение выражения "ложь", то параметр цикла не получает начального значения, тело цикла ни разу не выполняется и не переходим к шагу 2 (выходим из цикла).

2 шаг (и последующие).

2.1 Выполняются действия в теле цикла. Переходим к п.2.2.

2.2 Проверяется соотношение между текущим значением параметра цикла и его конечным значением. Вид соответствующего (проверяемого) логического выражения зависит от вида цикла:

Логическое выражение

для верхнего примера(to) ----- параметр цикла < "конечное значение"
 для нижнего примера (downto) ----- параметр цикла > "конечное значение"

Если это логическое выражение имеет значение "истина", то параметр цикла увеличится на 1 (для верхнего случая - to) или уменьшится на 1 (для нижнего случая - downto), после чего переход к пункту 2.1

Если это логическое выражение имеет значение "ложь", то параметр цикла значение не меняет и выполнение цикла завершится.

Рассмотрим два простых примера:

1)
 Var
 i: integer;
 begin
 i:=-2;
 for i:=0 to 4 do
 writeln(i);
 writeln(i);
 end.

2)
 Var
 i: integer;
 begin
 i:=-2
 for i:=5 to 4 do
 writeln(i);
 writeln(i);
 End.

выводится оператором
 writeln
 после цикла

1) Цикл будет выполняться пять раз. Выведет значение i : 0, 1, 2, 3, 4, 4. По выходу из цикла параметр м.б. равен последнему значению при котором последний раз выполнялось тело цикла.

2) Цикл не будет выполняться не разу потому, что начальные условия (см. шаг 1) не выполняются. В результате действия, включения в цикл будут пропущены Writeln(i) выведет (-2). Начальное значение параметр цикла получает после того, как цикл начинает выполняться.

3) for i:=4 to 4 do
 writeln(i);
 writeln(i);

Здесь на экран будет выведено 4
 4

В Турбо Паскале, начиная с 7-й версии, появились 2 псевдооператора, которые позволяют **менять ход выполнения цикла**:

For i := 1 to 10 do
 begin

...
 break;

continue; //к п.2.1 (см. выше)

...
 end;

если циклы вложенные (телом цикла является цикл)


Break - позволяет выполнить внезапный (одноуровневый) выход из цикла (в охватывающий его).

Continue выполняет следующее:


- прерывается текущее выполнение тела цикла при текущем значении параметра цикла
- изменяется значение параметра цикла на следующее (см. п.2.2 для описания работы цикла).
- начинается следующая итерация цикла (выполнение тела цикла со следующим значением параметра цикла).

Можно привести 2 Примера:

```
for i := 1 to 5 do
begin
  .....
  for j := 1 to 10 do
  begin
    .....
    break;
    .....
  end;
  оператор тела цикла по параметру i
  .....
end;
```

A blue curved arrow originates from the 'break;' statement and points to the 'end;' statement of the outer 'for i' loop, indicating that the loop is terminated.

```
for i := 1 to 5 do
begin
  write(i);
  Continue;
  writeln(i+2);
end;
```

A blue curved arrow originates from the 'Continue;' statement and points to the 'end;' statement of the 'for i' loop, indicating that the current iteration is skipped.

На экран будет выведено: 12345

Далее - 8. Совместимость и преобразование типов