Лекция 7. Алгоритмизация и программирование циклических алгоритмов

1. Примеры циклических алгоритмов

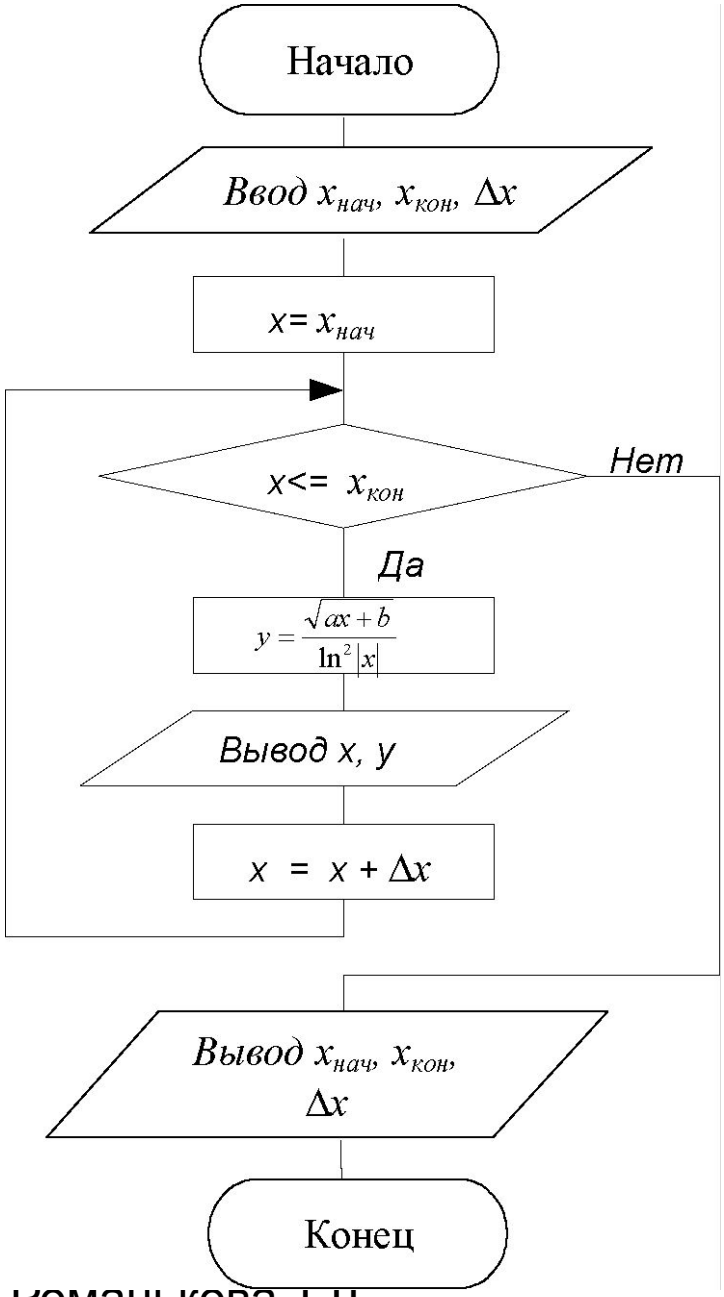
Циклический - алгоритм, в котором некоторая последовательность действий может выполняться несколько раз в зависимости от заданного условия.

Повторяющаяся последовательность действий (в программе операторов) называется телом цикла.

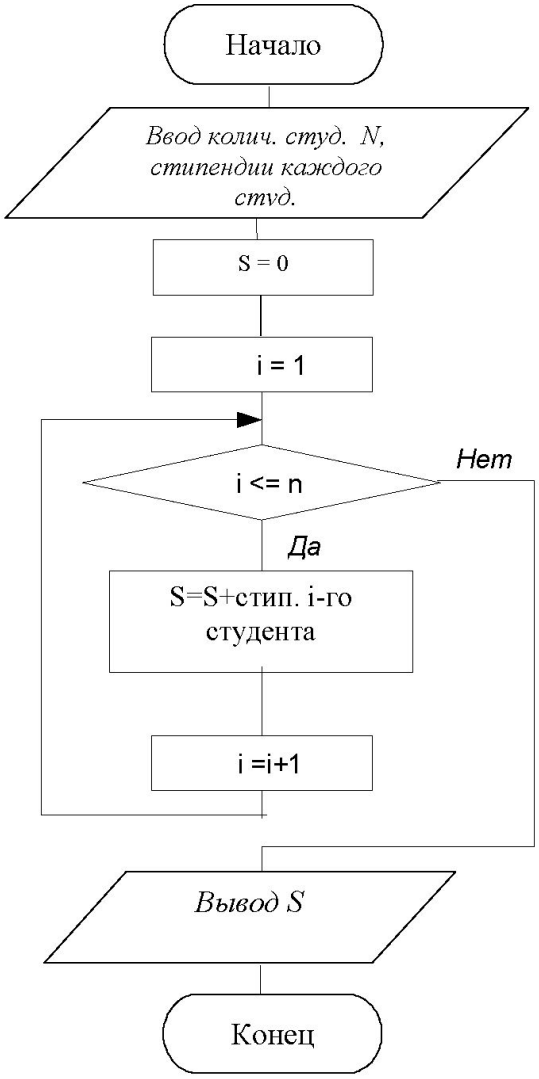
Условие, от которого зависит число повторений цикла, включает в себя по крайней мере одну переменную, которая наз. переменной цикла или **параметром** цикла.

Пример 1. Составить алгоритм вычисления функции для значений аргумента x, изменяющегося в интервале от $x_{нач}$ до $x_{кон}$ с шагом Δx, и заданных констант a и b. Такая задача называется задачей о **табулировании функции**.

$$y = \frac{\sqrt{ax + b}}{\ln^2 |x|}$$


Начало
Ввод $x_{нач}$, $x_{кон}$, Δx
$x = x_{нач}$
$x <= x_{кон}$
Нет
Да
$y = \frac{\sqrt{ax+b}}{\ln^2|x|}$
Вывод x, y
$x = x + \Delta x$
Вывод $x_{нач}$, $x_{кон}$, Δx
Конец

Пример 2. Определить стипендиальный фонд группы ТЭ-11.

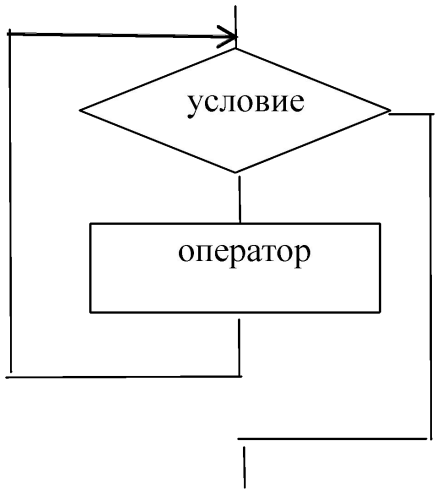


2. Операторы, необходимые для реализации циклического алгоритма.

Для реализации циклических алгоритмов используются операторы цикла:

while, for, repeat.

Оператор цикла while (цикл с предусловием).



while <условие> do
<оператор>;

<Условие> – выражение логического типа.

Оператор while организует повторение одного оператора до тех пор, пока условие истинно.

Если в тело цикла нужно включить несколько операторов, то эти операторы нужно объединить в один составной, заключив их в операторные скобки: begin и end.

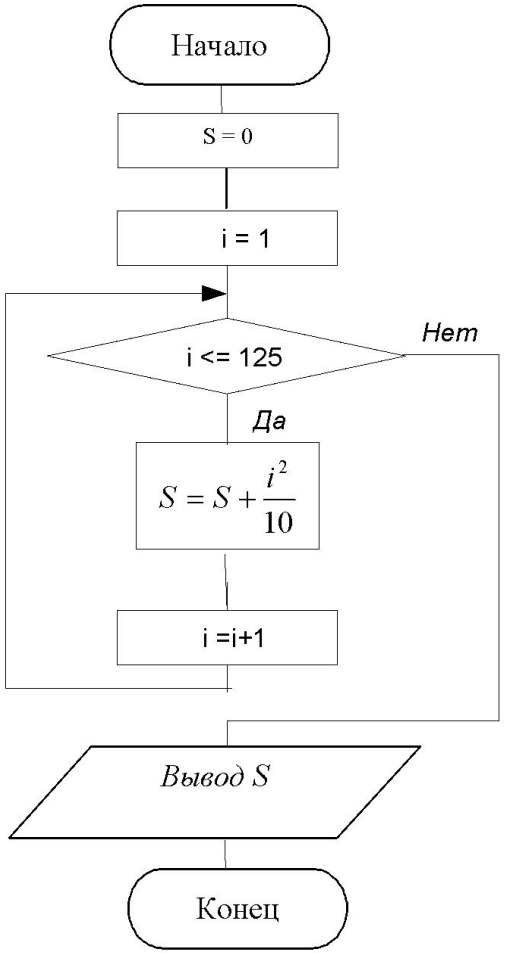
Оператор **while** выполняется следующим образом:

- Сначала вычисляется значение выражения в условии.
- Если значение выражения условия равно True (условие выполняется), то выполняется оператор, стоящий после слова do (тело цикла). После этого снова проверяется выполнение условия. Если условие выполняется, то тело цикла выполняется еще раз. И т.д.
- Если значение выражения равно False (условие не выполняется), то на этом выполнение оператора while завершается, и происходит переход к первому после while оператору.

Для того чтобы операторы тела цикла while были выполнены хотя бы один раз, необходимо, чтобы перед выполнением оператора while значение выражения условия было истинно.

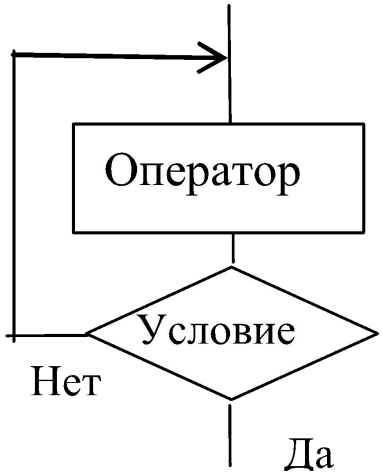
Для того чтобы цикл завершился, нужно, чтобы оператор, стоящий после do, влиял на значение выражения условия (изменял значения переменных цикла).

Пример 3. Ниже приведен алгоритм вычисления $\sum_{i=1}^{125} \frac{i^2}{10}$ и его реализация с помощью оператора **while**




```
S:=0;
i:=1;
While i<=125 do
  Begin
    S:=S+sqr(i)/10;
    i:=i+1
  End;
```

Оператор цикла repeat (цикл с постусловием).

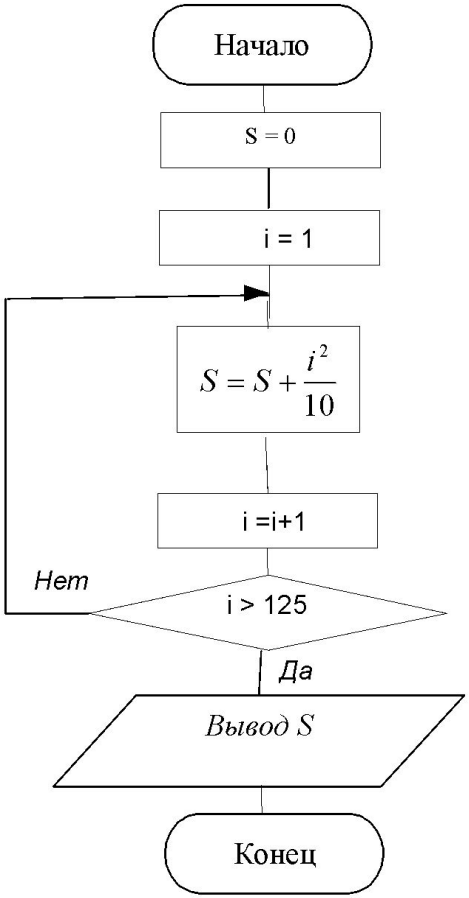


Repeat
<оператор 1>;
<оператор 2>;
…
<оператор n>
Until <условие>;

Оператор ***Repeat*** выполняется следующим образом:

- Сначала выполняются находящиеся между repeat и until операторы тела цикла.
- Затем вычисляется значение выражения условия. Если условие ложно (значение выражения условие равно False), то операторы тела цикла выполняются еще раз и т.д.
- Если условие истинно (значение выражения условия равно True), то выполнение цикла прекращается.

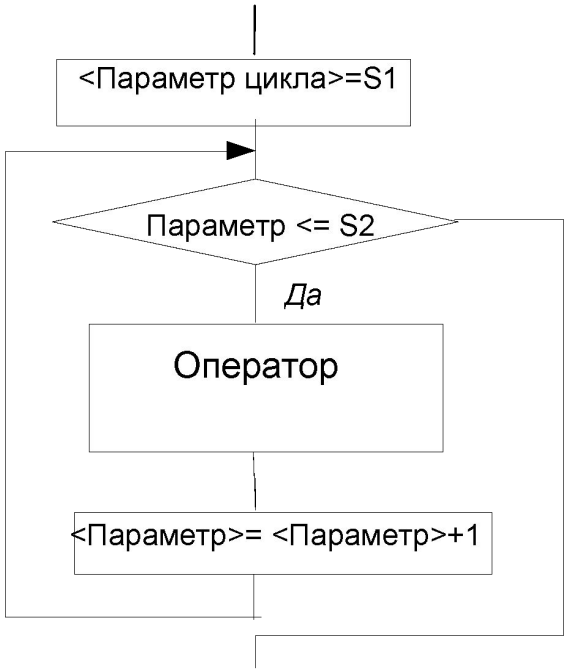
Пример 4. Ниже приведен алгоритм вычисления $\sum_{i=1}^{125} \frac{i^2}{10}$ и его реализация с помощью оператора **repeat.**




```
S:=0;
i:=1;
repeat
   S:=S+sqr(i)/10;
   i:=i+1
Until i>125;
```

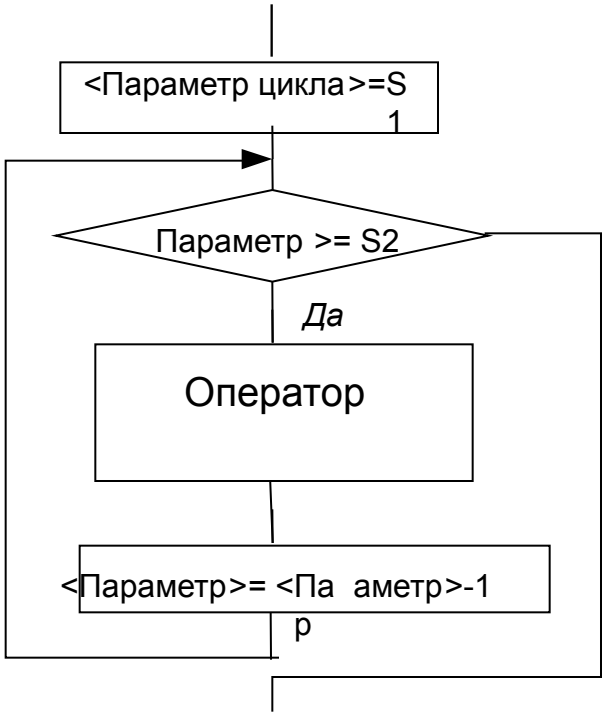
Оператор цикла for

Оператор for используется в том случае, если число повторений цикла заранее известно.



for <Параметр> : = S1 to S2 do
<оператор>;

После каждого выполнения оператора параметр цикла автоматически увеличивается на 1



for <Параметр> : = S1 downto S2 do
<оператор>;

После каждого выполнения оператора параметр цикла автоматически уменьшается на 1

- <Параметр> — переменная-счетчик числа повторений тела цикла;
- S1-- выражение, определяющее начальное значение счетчика цикла;
- S2 — выражение, определяющее конечное значение счетчика цикла.

Параметр цикла, а также выражения S1 и S2 должны быть целого типа**.**

Если в тело цикла нужно включить несколько операторов, то эти операторы нужно объединить в один составной, заключив их в операторные скобки: begin и end.

Для примера 3 фрагмент программы, реализующей алгоритм с помощью оператора **for,** выглядит так:

```
S:=0;
For i:=1 to 125 do
    S:=S+sqr(i)/10;
```

3.Визуальный компонент **StringGrid (**страница **Additional)**.

Предназначен для создания и редактирования таблиц, в ячейках которых располагаются произвольные текстовые строки.

Таблица делится на две части – фиксированную и рабочую.

Фиксированная служит для выделения заголовков строк и столбцов. Обычно занимает левый столбец и верхнюю строку таблицы.

Рабочая часть – остальная часть таблицы.

Свойства:

Самое главное –

Cells – набор ячеек, содержащих редактируемый текст.

Каждая ячейка определяется парой чисел: номером столбца и номером строки, на пересечении которых она находится.

Нумерация начинается с нуля. Ячейка коллекции **Cells** имеет тип String. Для обращения к конкретной ячейке используют составное имя, например

StringGrid1.Cells[1,2]

ячейка, расположенная в 1-м столбце 2-й строки.

FixedCols – количество фиксированных столбцов.

FixedRows– количество фиксированных строк.

ColCount - количество столбцов

RowCount - количество строк

Visible определяет отображение таблицы в окне при выполнении программы(true – видимая, false – невидимая).

Опция **goEditing** свойства **Options** определяет возможность редактирования текста в ячейках таблицы (true – можно редактировать, false – нельзя.

4. Программирование решения задачи о табулировании функции.

Порядок решения задачи:

- разработать проект формы:

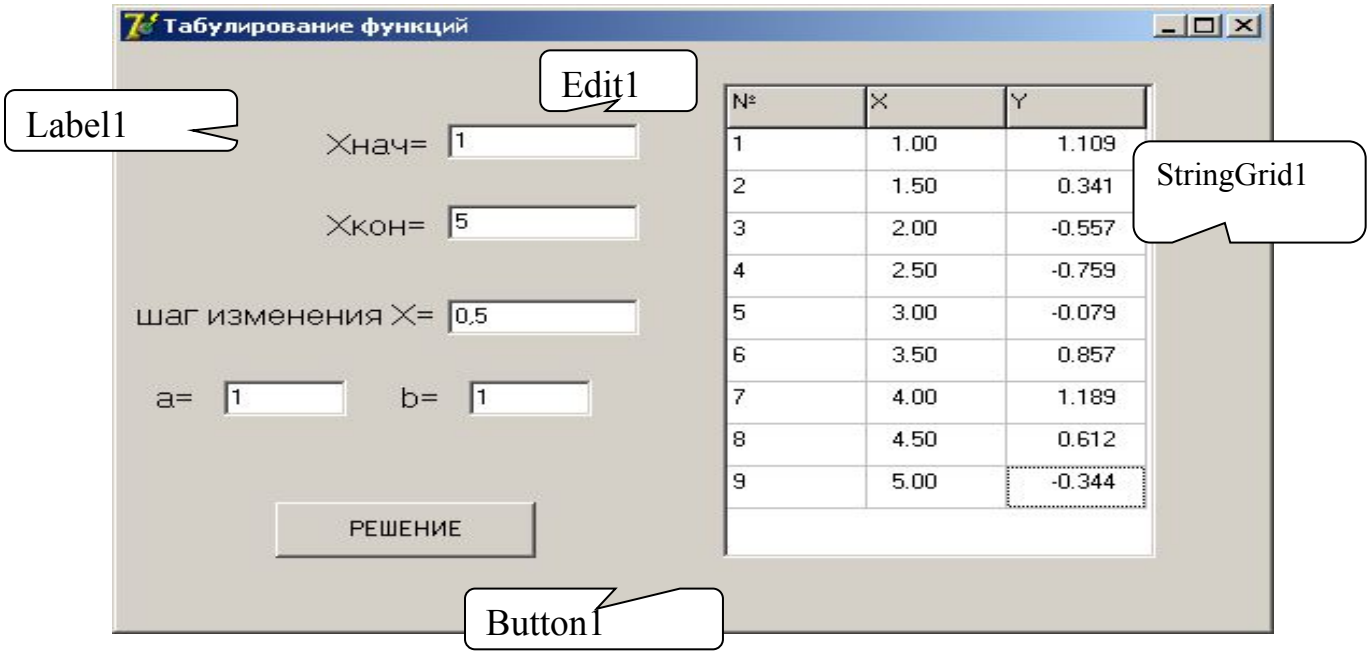


Таблица свойств компонентов:

Элемент интерфейса	Компонент	Свойства	Значение свойства
Главное окно программы	Form 1	Caption Position	Табулирование функции poScreenCenter
Комментарий «Хнач=»	Label 1	Caption	Хнач=
Комментарий «Хкон=»	Label 2	Caption	Хкон=
Комментарий «шаг изменения Х=»	Label 3	Caption	шаг изменения Х=

Поле ввода значения Хнач	Edit1	Text	Очистить
Поле ввода значения Хкон	Edit2	Text	Очистить
Поле ввода шага изменения	Edit3	Text	Очистить
Поле ввода параметров a, b	Edit4, Edit5	Text	Очистить
Кнопка «Решение»	Button1	Caption Hint ShowHint	Решение Вычисление значения функции True
Таблица для ввода значений функции	StringGrid1	FixedCols FixedRows ColCount Visible	0 1 3 False

- разместить все ВК на форме и установить их свойства с помощью Инспектора объектов.
- разработать алгоритм решения задачи: см. пример 1.
- Подобрать для переменных имена, сформированные в соответствии с правилами языка программирования

Например:

Имя переменной в условии задачи	Имя переменной в программе	Тип	Комментарии
$x_{нач}$	*хп*	вещественный	Начальное значение аргумента
$x_{кон}$	*хк*	вещественный	Конечное значение аргумента
Δx	*dx*	вещественный	Шаг изменения аргумента

x	*x*	вещественный	Текущее значение аргумента
y	*y*	вещественный	Значение функции
a	*a*	вещественный	Параметр функции
b	*b*	вещественный	Параметр функции
	i	целый	Порядковый номер строки таблицы
	Sx, sy	Строковый	Значения аргумента и функции в виде форматированной строки

- создать процедуру, которая будет выполняться при нажатии кнопки. Для этого нужно выполнить двойной щелчок мышью по кнопке.

- В окне кода набрать текст процедуры, реализующий составленный алгоритм:

```
procedure TForm1.Button1Click(Sender: TObject);

  var
    x,xn,xk,dx,y,a,b:real;
    sx,sy:string;
    i:integer;
begin
  xn:=strtofloat(edit1.Text); // ввод нач. значения арг.
  xk:=strtofloat(edit2.Text); // ввод кон. значения арг.
  dx:=strtofloat(edit3.Text); // ввод шага
  a:=strtofloat(edit4.Text); // ввод a
  b:=strtofloat(edit5.Text); // ввод b
```

```
Stringgrid1.Visible:=true;   // сделать видимой таблицу
stringgrid1.Cells[0,0]:='№';  //   вывод
stringgrid1.Cells[1,0]:='X';  //   шапки
stringgrid1.Cells[2,0]:='Y';  //   таблицы
stringgrid1.RowCount:=round((xk-xn)/dx)+2; { Определение
  количества строк  в таблице, round - функция округления}
 x:=xn;
 i:=1;
while x<=xk do
  begin
     y:=sqrt(a*x+b)/sqr(ln(abs(x)));
  str (x:7:2,sx);str(y:8:3,sy);// преобразование значений
                                //  x и y к строковому типу
  stringgrid1.Cells[0,i]:=inttostr(i);// вывод номера строки
                                      // таблицы
```

```
stringgrid1.Cells[1,i]:=sx;  // вывод текущего значения x
stringgrid1.Cells[2,i]:=sy; // вывод текущего значения y
 i:=i+1;
 x:=x+dx

  end;
end;
```

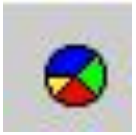
- Сохранить полученную программу командой Save All **в отдельной папке**

- Запустить программу на выполнение командой Run или нажатием клавиши F9 и проверить работу программы с помощью заранее подготовленных тестов.

Например, при *хнач* = 2, *хкон* = 4, *Δx* = 0.5, a = 1, b = 1
ожидаемый результат

№	x	y
1	2	3.605
2	2.5	2.228
3	3	1.657
4	3.5	1.352
5	4	1.164

5. Построение графика функции
5.1. Визуальный компонент **TChart- диаграмма. (страница Additional)**



Предназначен для графического представления числовых данных.

Основное свойство – **SeriesList -** пронумерованный набор серий, отображающих наборы чисел в графическом виде (нумерация серий начинается с нуля). Чтобы построить один график, нужно создать одну серию.

Схема использования компонента **TChart:**

- Поместить компонент на форму
- Дважды щелкнуть на нем левой кнопкой мыши для вызова многостраничного окна редактора
- На вкладке **Series** щелкнуть по кнопке **Add** для добавления серии и выбрать подходящий тип диаграммы
- Установить другие свойства компонента, например, на вкладке **Titles** набрать заголовок, на вкладке **Legend** убрать легенду, сняв флажок в переключателе **Visible**, и т.д.

- В процедуре, генерирующей данные для графика, очистить серию с помощью метода **Clear:**

Например, **`Chart1.SeriesList[0].Clear;`**

- добавлять точку к серии с помощью метода **AddXY:**

<Имя компонента>.<серия>.AddXY(<абсцисса>,<ордината>);

Например, точка с координатами (x,y) на графике отображается с помощью оператора

Chart1.SeriesList[0].AddXY(x,y);

5.2 Построение графика функции из примера 1

Для построения графика функции нужно:

- на форму поместить компонент класса **TChart,**
- вызвать окно редактирования компонента, добавить серию с типом **Line,**
- на вкладке **Legend** убрать легенду, на вкладке **Titles** набрать заголовок,
- на вкладке **3D** убрать флажок переключателя **3 Dimensions,**
- при необходимости изменить другие свойства компонента.

Процедуру, которая будет выполняться при нажатии кнопки, нужно дополнить операторами очистки серии и отображения на графике очередной точки:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  ...  {Описание переменных}
begin
  ... {Ввод исходных данных, вывод шапки таблицы}
  Chart1.SeriesList[0].Clear;
  x:=xn;
  i:=1;
  while x<=xk do
    begin
      ...{Вычисление текущего значения функции и
           вывод строки таблицы}
      Chart1.SeriesList[0].AddXY(x,y);
      i:=i+1;
      x:=x+dx
    end;
end;
```