Лабораторная работа № 3

Табулирование составной функции

Задание: Составить блок-схему и программу для заполнения таблицы значений функции $y = f(x)$ на отрезке с указанным шагом изменения аргумента. Вид функции задается в лабораторной работе № 1. Значение функции выводить с точностью до тысячных долей.

Результат вывести в следующем виде:

```
----------------------------------------
!          X        !       y = f(x)       !
----------------------------------------
```

Теоретический материал

Реализация подобного рода задач требует повторения действия (или нескольких действий) несколько раз, т. е. использования циклов.

В языке Pascal имеется три вида циклических структур: цикл с параметром `for`, цикл с предусловием `while` и цикл с постусловием `repeat..until`.

Цикл с параметром используется тогда, когда заранее известно количество повторов тела цикла. Число повторений здесь задается переменной, называемой параметром цикла, или управляющей переменной.

Данный оператор может быть представлен в двух форматах:

```
for <параметр цикла> := <S1> to <S2> do <тело цикла>;
for <параметр цикла> := <S1> downto <S2> do <тело цикла>;
```

где `S1` и `S2` – выражения, определяющие соответственно начальное и конечное значения параметра цикла.

Тело цикла может содержать один оператор или несколько операторов, заключенных в операторные скобки `begin .. end` (составной оператор). **Если тело цикла – простой оператор**, **операторные скобки не пишутся!**

При первом обращении к оператору `for` вначале вычисляются выражения `S1`, `S2` и осуществляется присваивание `<параметр цикла>:=S1`.

После этого циклически повторяются следующие действия.

1. Проверяется условие `<параметр цикла>:<=S2`.
2. Если условие выполнено, то оператор for продолжает работу (выполняется оператор в теле цикла), если условие `<параметр цикла>:<=S2` не выполнено, то оператор `for` завершает работу, и управление в программе передается на оператор, следующий за циклом.
3. Значение управляющей переменной изменяется на +1 (`to`) или -1 (`downto`) и далее с п. 1. Обратите внимание, что шаг изменения управляющей переменной – единица.

На использование управляющей переменной (параметра цикла) в цикле `for` налагаются следующие ограничения.

1. В качестве параметра должна использоваться простая переменная, описанная в текущем блоке.
2. Управляющая переменная должна иметь дискретный тип (как правило, `integer`, реже `Char`, `Boolean`).
3. Начальные и конечные значения диапазона должны иметь тип, совместимый с типом управляющей переменной. При этом допустим любой скалярный тип, кроме вещественного.
4. В теле цикла не рекомендуется явное изменение значения управляющей переменной (например, оператором присваивания), т. к. параметр цикла изменяется автоматически.

Цикл с предусловием используется тогда, когда число повторений заранее неизвестно – надо выполнять цикл, пока не произойдет некоторое событие (пользователь нажмет на кнопку, точность вычислений уложится в заданный порог и т. д.). В операторе `while` проверка условия выполнения тела цикла производится в самом начале оператора.

Формат записи:

```
while <условие продолжения цикла> do
<тело цикла>;
```

Условие продолжения цикла – булевское (логическое) выражение, тело цикла – простой или составной оператор. **Если тело цикла – простой оператор, операторные скобки не пишутся!**

Особенность данного вида цикла состоит в том, что значение выражения условия вычисляется перед каждым выполнением тела цикла. Если результат равен True, тело цикла выполняется и снова вычисляется выражение условия. Если результат равен False, происходят выход из цикла и переход к первому после `while` оператору.

Еще один важный момент, о котором нельзя забывать программисту – зацикливание (бесконечное выполнение цикла). Чтобы не происходило зацикливание программы, нужно, чтобы в теле цикла присутствовал либо оператор, изменяющий значение, проверяемого в условии продолжения цикла, либо оператор безусловного выхода из цикла (`break`).

Цикл с постусловием используется в тех случаях, когда заранее известно, что тело цикла повторится, по меньшей мере, один раз. Оператор цикла `repeat..until` аналогичен оператору `while`, но отличается от него, во-первых, тем, что условие проверяется после очередного выполнения операторов тела цикла и таким образом гарантируется хотя бы однократное выполнение цикла, а во-вторых, цикл завершается, когда условие, стоящее после `until` становится истинным, т. е. значение логического выражение – True (в операторе `while` цикл завершается, когда условие принимает значение False).

Формат записи:

```
repeat
   <оператор;>
   …
   <оператор>
until <условие окончания цикла>;
```

Цикл работает следующим образом: выполняется тело цикла (операторы, заключенные между словами `repeat` и `until`), затем проверяется условие окончание цикла, если оно пока еще не выполнилось, то тело цикла выполняется вновь, затем проверяется условие, и т. д. Когда условие, наконец, станет истинным, цикла завершится, и далее будет выполняться следующий за циклом оператор.

Следует заметить, что тело цикла в данном случае может содержать как один, так и несколько операторов, однако, операторные скобки здесь не пишутся, т. к. `repeat` обозначает начало цикла, `until` – конец цикла.

Так же как и в случае с циклом `while`, при программировании операторов тела цикла следует обеспечить влияние, по крайней мере, одного из операторов тела цикла на значение условия, иначе цикл будет выполняться бесконечно.

Пример выполнения лабораторной работы

Для примера рассмотрим функцию $y = x^3 + \frac{x}{\sin x}$, аргумент которой изменятся в интервале [-π; π] с шагом π/5.

Составим блок-схему алгоритма (рис. 3.1).

Программную реализацию начинаем с описания переменных. Как видно из блок-схемы алгоритма, в программе понадобятся две переменные: x – для хранения значения аргумента, y – для хранения значения функции. Поскольку обе переменные могут иметь дробные значения, описываем их вещественным типом. Из соображений минимизации времени выполнения программы целесообразно шаг приращения аргумента x задать как величину постоянную, значение которой будет вычисляться единожды при описании константы. Заметим, что в языке Pascal имеется стандартная константа Pi, которую мы используем при вычислении шага.

```
Const h=Pi/5;
Var x,y:real;
```

Тело программы начинается с уже знакомого информационного блока и оператора, печатающего «шапку» будущей таблицы. Затем задается начальное значение аргумента функции, которое определяется нижней границей изменения x.

```
 x:=-Pi;
```

Далее организуется цикл, в ходе выполнения которого будут вычисляться и печататься значения аргумента и функции. Поскольку в данном случае заранее неизвестно число повторений цикла, будем использовать цикл с предусловием, в

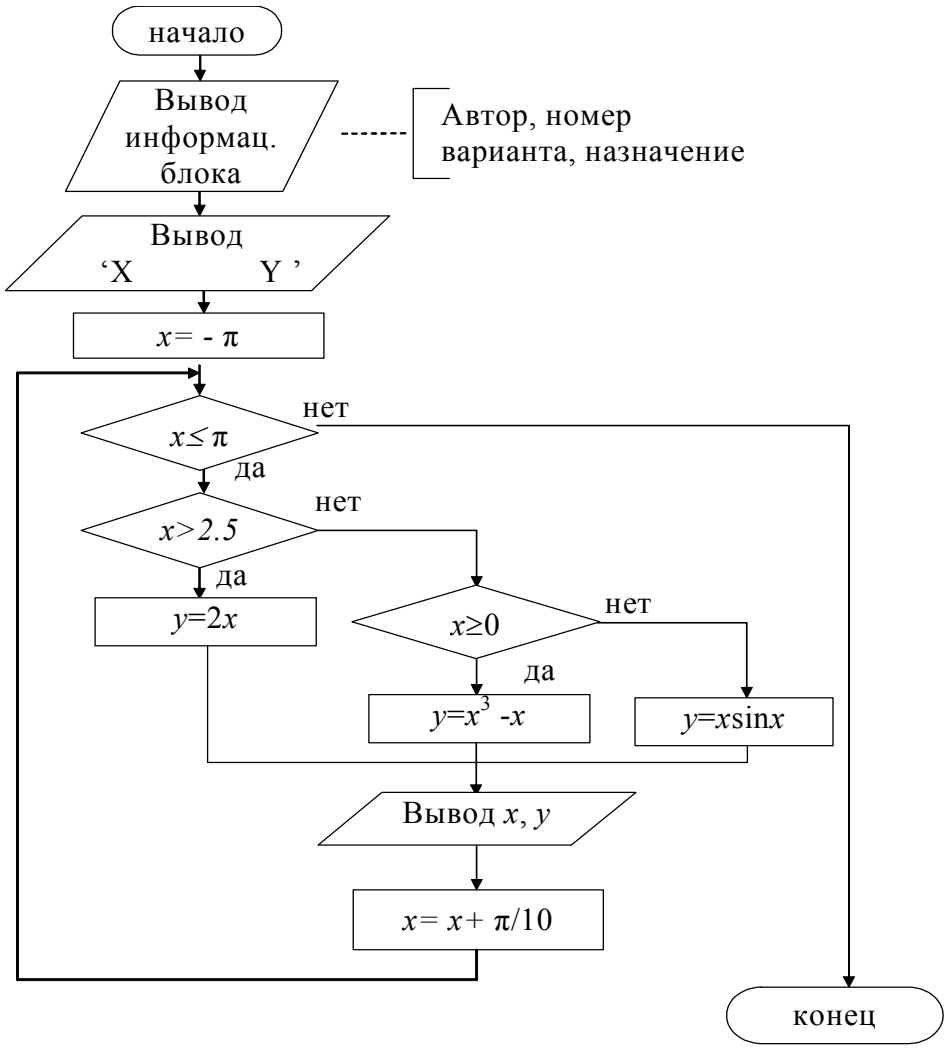

Рис. 3.1. Блок-схема задачи табулирования функции

котором условием продолжения цикла будет $x \le \pi$, а телом цикла – составной оператор, включающий группу операторов вычисления значения составной функции (см. лабораторную работу № 1) и оператор, увеличивающий текущее значение аргумента на величину шага.

```
While x<=Pi do
Begin
  If x>2.5 then
    Y:=2*x
  Else
    If x>=0 then
       y:=sqr(x)*x-x  {sqr(x) – возведение x в квадрат}
    Else
       Y:=x* sin(x);
  Writeln(x:7:3,y:10:3);
  x:=x+h
end;
```

После выхода из цикла остается задержать результаты работы на экране с помощью пустого оператора ввода и завершить программу.

```
Readln
End.
```

Теперь соберем все фрагменты в единую программу.

```
Program Lab3;
Const h=Pi/5;
Var x,y:real;
Begin
  writeln;
  writeln('       Автор – Иванов И.П., студент гр. ИСЭд-11');
  writeln('       Вариант № 100');
  writeln(' Программа для заполнения таблицы значений функции');
  writeln('   |2*x                  x>2.5');
  writeln('Y= |x^3-x            0<=x<=2.5');
  writeln('   |x*(sin(x))           x<0');
  writeln('на отрезке [-Pi; Pi] с шагом Pi/5');
  writeln;
  x:=-Pi;
  While x<=Pi do
  Begin
    If x>2.5 then
      Y:=2*x
    Else
      If x>=0 then
         y:=sqr(x)*x-x
      Else
         Y:=x* sin(x);
    Writeln(x:7:3,y:10:3);
    x:=x+h
  end;
  Readln
  End.
```

Таблица 3.1

Варианты заданий

№ вар.	Функция	Интервал изменения аргумента	Шаг изменения аргумента
1.	$y = \begin{cases} x - 2 & x > 2.5 \\ 1 + x^2 & 0 \le x \le 2.5 \\ x \ln\lvert\cos(x)\rvert & x < 0 \end{cases}$	$[-\pi; \pi]$	$\pi/10$

Продолжение таблицы 3.1

2.	$y = \begin{cases} \sin(2.3x - 1) \\ 1 - 3\ln\lvert 1 - x\rvert \\ \dfrac{x^2}{2 - x} \end{cases}$	x > 2.5 0 <= x <= 2.5 x < 0	[-π/5; 9π/5]	π /3
3.	$y = \begin{cases} \sqrt{(tg(x^2 - 1))} \\ -2x \\ e^{\cos(x)} \end{cases}$	x > 1 0 ≤ x ≤ 1 x < 0	[-1; 1.5]	0.5
4.	$y = \begin{cases} x^2 - 3 + 2.5x^2 \\ e^x + 5 + \cos(0.001x) \\ x^2 \end{cases}$	x>12.5 0 ≤ x ≤12.5 x < 0	[-5; 10]	0.55
5.	$y = \begin{cases} 1 + \sqrt{\lvert\cos(x)\rvert} \\ x + 1 \\ 1 - x^2 \end{cases}$	x >1 -0.5 ≤ x ≤ 1 x < -0.5	[-1.5; 1.5]	0.25
6.	$y = \begin{cases} 2.5 \cdot x^3 + 6 \cdot x^2 - 30 \\ x + 1 \\ x \end{cases}$	x > 1.5 0 ≤ x ≤ 1.5 x < 0	[-2; 3]	0.5
7.	$y = \begin{cases} 1 + x \\ e^{-x} \\ \cos(x) \end{cases}$	x > 14.5 3 ≤ x ≤ 14 .5 x <3	[-1; 15]	1
8.	$y = \begin{cases} \ln\lvert 1 + x\rvert \\ e^{-x} \\ \cos(x) \end{cases}$	x > 3.8 2.8 ≤ x ≤ 3.8 x < 2.8	[0; 5]	0.5
9.	$y = \begin{cases} 1 + \sqrt{\cos(x)} \\ x + 1 \\ 1 - x^2 \end{cases}$	x > 4 0 ≤ x ≤ 4 x < 0	[-1; 4.5]	0.25
10.	$y = \begin{cases} e^{-(x + 8)} \\ 1 \\ \dfrac{x}{5} \end{cases}$	x > 3.61 0 ≤ x ≤ 3.61 x < 0	[-π; 2π]	π /5

Продолжение таблицы 3.1

11.	$y = \begin{cases} x & \\ 2x^2\sqrt{\lvert\cos(2x)\rvert} & \\ e^{-\cos(3x)} & \end{cases}$	x > 1.5 0 ≤ x ≤ 1.5 x < 0	[-1; 3]	0.5
12.	$y = \begin{cases} 1-\sqrt{\cos(2x)} & \\ x^2 - x & \\ 1 + x^2 & \end{cases}$	x > 2.5 1 ≤ x ≤ 2.5 x < 1	[0; 3]	0.3
13.	$y = \begin{cases} 2x & \\ 1-\ln\lvert 1-x^2\rvert & \\ e^{-x} & \end{cases}$	x > 4.5 0 ≤ x ≤ 4.5 x < 0	[-0.5; 5]	0.5
14.	$y = \begin{cases} \sqrt{\ln(x^2-1)} & \\ -2x^3 & \\ e^{\sin(x)} & \end{cases}$	x > 2 0 ≤ x ≤ 2 x < 0	[-π/2; π]	π /5
15.	$y = \begin{cases} 1-\dfrac{2x^3}{1-x^2} & \\ \sqrt{\cos(2x-1)} & \\ e^{-\cos(2x)} & \end{cases}$	x > 3.5 0 ≤ x ≤3.5 x < 0	[-0.5; 4.5]	0.25
16.	$y = \begin{cases} x+1 & \\ 1-x^5 & \\ x+\ln\lvert\sin(x)\rvert & \end{cases}$	x > 2.5 0 ≤ x ≤ 2.5 x < 0	[-π; 2π]	π /5
17.	$y = \begin{cases} x-2 & \\ 1+x^2 & \\ x\ln\lvert\cos(x)\rvert & \end{cases}$	x > 2.5 0 ≤ x ≤ 2.5 x < 0	[-π/2; 2π]	π /4
18.	$y = \begin{cases} 1+3x & \\ e^{-2x} & \\ \cos(2x) & \end{cases}$	x > 4.5 1 ≤ x ≤4.5 x < 1	[-π/2; 2π]	π /5
19.	$y = \begin{cases} \sqrt{\lvert tg(x^2-1)\rvert} & \\ -2x & \\ e^{\cos(x)} & \end{cases}$	x > 4 0 ≤ x ≤ 4 x < 0	[-2; 5]	0.5

Окончание таблицы 3.1

20.	$y = \begin{cases} e^{(x+2)} & x > 1 \\ 1 - 2x & -1 \le x \le 1 \\ -\dfrac{2x^3 - 3}{5} & x < -1 \end{cases}$	[0.8; 2.5]	0.1
21.	$y = \begin{cases} 1 - x^3 & x > 2 \\ -2x & 0 \le x \le 2 \\ e^{\cos(x)} & x < 0 \end{cases}$	[-1; 2.5]	0.25
22.	$y = \begin{cases} 1 + \sqrt{\lvert tg(x) - 1 \rvert} & x < -3.14 \\ x & -3.14 \le x \le 3.14 \\ 1 + x^2 & x > 3.14 \end{cases}$	[-9π/5; 9π/5]	π /5
23.	$y = \begin{cases} \dfrac{1}{\ln(x^3)} & x > 4.5 \\ 2x + 0.1 & 0 \le x \le 4.5 \\ 3 & x < 0 \end{cases}$	[-0.5; 5]	0.25
24.	$y = \begin{cases} x^2 - 3 + 2.5x^3 & x > 2 \\ e^{(x)} + 5 + \cos(0.001x) & -1 \le x \le 2 \\ \ln\lvert tg(2x) \rvert - 1 & x < -1 \end{cases}$	[-2.5; 2.5]	0.5
25.	$Y = \begin{cases} \cos(2.3x - 1) & x > 5.5 \\ 1 - 3\ln(1 + x) & 0 \le x \le 5.5 \\ \dfrac{x^2}{2 - x} & x < 0 \end{cases}$	[-1; 8]	0.5

Лабораторная работа № 4

Табулирование функции и ее разложения в сумму ряда

Задание: Составить блок-схему и программу табулирования двух функций `S` и `Y` в заданном диапазоне изменения аргумента `x`. Здесь `n` - число слагаемых суммы `S`.

Результат табулирования вывести в форме следующей таблицы:

```
-------------------------------------------------------------
!     X       !          y = f(x)            !        S       !
-------------------------------------------------------------
```