ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Python. Проектирование алгоритмов и программ со структурой вложенных циклов

Методические указания к лабораторной работе

10

Рязань 2017

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Python. Проектирование алгоритмов и программ со структурой вложенных циклов

Лабораторная работа №10

Методические указания к лабораторной работе

Рязань 2017

УДК 004.432

Python. Проектирование алгоритмов и программ со структурой вложенных циклов: методические указания к лабораторной работе / Рязан. гос. радиотехн. универ.; Сост.: А.Н. Пылькин, Н.Н. Степанов, Н.А. Тярт. – Рязань, 2017 г. – 16 с.

С позиций структурного программирования рассмотрены основные практические приемы проектирования алгоритмов и программ со структурой вложенных циклов. В качестве практических примеров приведено решение таких задач, как табулирование функций двух переменных, вычисление кратных сумм и произведений, а также численное вычисление определенных интегралов с автоматическим шагом интегрирования, что обеспечивает требуемую точность вычислений.

Ил.: 6. Бибилиогр.: 4 назв.

Печатается по решению Научно-методического совета Рязанского государственного радиотехнического университета.

Рецензент: кафедра информатики, информационных технологий и защиты информации ФГБОУ ВО «Липецкий государственный педагогический университет им. П.П. Семенова-Тян-Шанского» (зав. каф., к.т.н., доц. Скуднев Д.М.).

Цель работы

Получение практических навыков проектирования алгоритмов и программ со структурой вложенных циклов.

Исходные определения

В вычислительной технике достаточно часто встречаются задачи, при решении которых необходимо проектировать программы с несколькими циклами, вложенными один в другой.

Вложенным циклом называют любой цикл, содержащий внутри себя один или несколько циклов. Цикл, охватывающий другие циклы, называют *внешним,* а остальные – *внутренними* циклами.

Правила организации внешнего и внутренних циклов одинаковы (как и для простого цикла). *Внутренний цикл должен полностью располагаться в теле внешнего цикла.*

Циклы, образующие вложенный цикл, условно разбивают на уровни вложения: внешний цикл – уровень 0, первый внутренний цикл – уровень 1, второй внутренний цикл – уровень 2 и т.д. При программировании каждого нового уровня расходуется определенный объем памяти для хранения параметров цикла, поэтому глубина вложения сложной циклической структуры ограничивается объемом имеющейся памяти. В программе на каждом уровне может быть использован любой из операторов цикла языка (с постусловием, предусловием, параметром).

Параметры циклов разных уровней изменяются не одновременно. Сначала все свои значения принимает поочередно параметр самого внутреннего цикла (при неизменных значениях параметров внешних циклов). После этого изменяется на один шаг параметр следующего по рангу цикла и снова все свои значения изменяет параметр цикла наибольшего уровня вложения. Этот процесс происходит до тех пор, пока параметры циклов всех уровней не примут все возможные значения.

Если во вложенном цикле реализованы две циклические структуры с параметрами $x = x_0(h_x)x_n$ и $y = y_0(h_y)y_n$, то общее число повторений тела внутреннего цикла

$$N = N_x * N_y,$$

где

$$N_x = \left[\frac{x_n - x_0}{h_x}\right] + 1; \quad N_y = \left[\frac{y_n - y_0}{h_y}\right] + 1.$$

Табулирование функций двух переменных

Рассмотрим функцию $z = f(x,y)$ от двух переменных x и y. Обе переменные могут выступать в качестве параметров циклов при табулировании функции.

В процессе проектирования алгоритма и программы не имеет значения, по какому параметру организовать внутренний и внешний циклы. Выберем в качестве внешнего параметра переменную x. Алгоритм табулирования функции по параметру x показан на рис. 1. Детализация тела цикла по параметру x – ТЦ$_x$ – реализуется в виде циклической структуры с параметром y. В результате получаем алгоритм табулирования функции $z = f(x,y)$ от двух переменных (рис. 2).

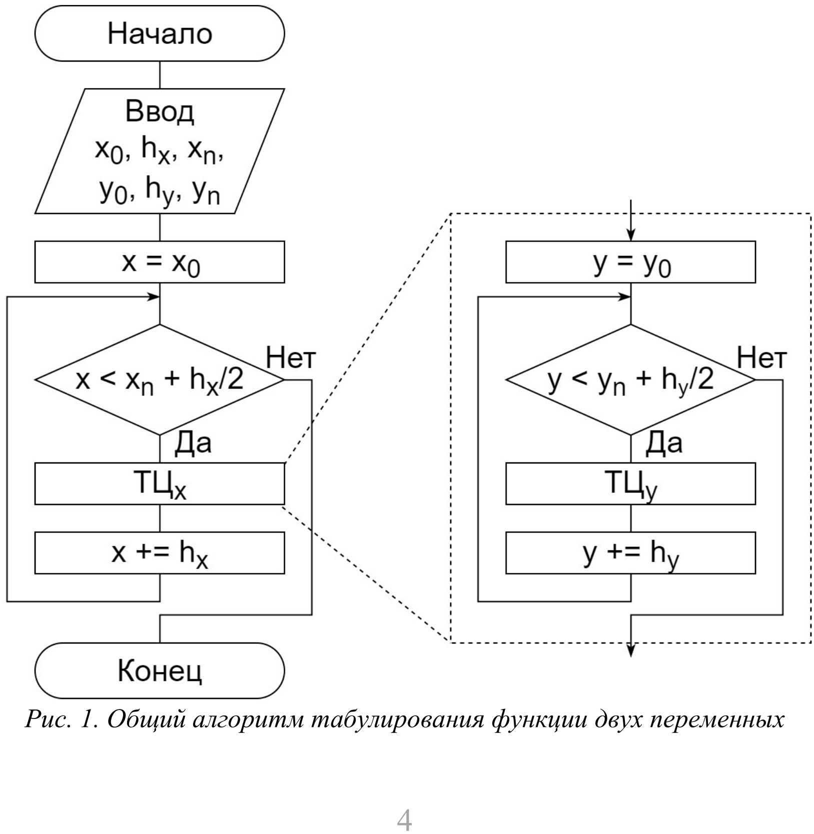


Рис. 1. Общий алгоритм табулирования функции двух переменных

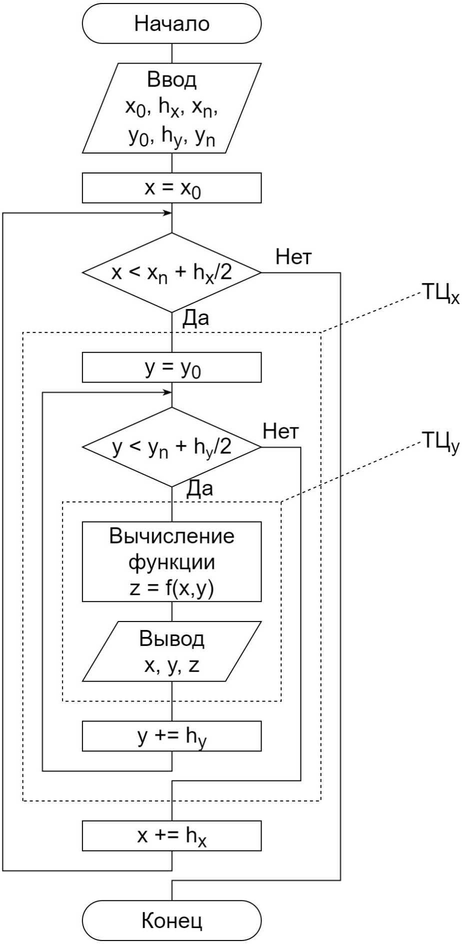


Рис. 2. Детализированный алгоритм табулирования функции двух переменных

Фрагмент программы, который схематично показывает структуру вложенных циклов, имеет следующий вид:

```
# ...
x0 = 1
y0 = 1
hx = 1
hy = 1
```

```
xn = 10
yn = 10

z = 0
x = x0
while x < (xn + hx / 2):
    y = y0
    while y < (yn + hy / 2):
        # ...
        # вычисление фукнции z = f(x, y)
        # ...
        print("z = %.2f \t x = %.2f \t y = %.2f"
              % (z, x, y))
        y += hy
    x += hx
# ...
```

Вычисление кратных сумм и произведений

В процессе вычисления кратной суммы требуется организовать вложенный цикл, который позволяет провести суммирование отдельных слагаемых.

Пусть необходимо вычислить значение функции

$$y = \frac{x + 1{,}5}{3{,}75} \sum_{k=0}^{10} \sum_{n=0}^{5} (k + n) x^{k+n}$$

при некоторых значениях аргумента $x = x_0(h_x)x_n$.

Проектирование алгоритма решения этой задачи начинаем с составления циклической структуры с постусловием, которая решает в общем виде задачу табулирования функции $y = f(x)$ (рис. 3).

Значение двойной суммы вычислим с помощью вложенного цикла по параметрам k и n, являющимися одновременно переменными суммирования. В качестве начального значения для суммы S берем нулевое.

Оба цикла *(k* и *n)* являются циклическими структурами с заголовком и целиком расположены внутри тела цикла по параметру *x*. Следует обратить внимание на то, что задание начальных значений переменных внутреннего цикла производится непосредственно *перед* входом *в данный* цикл.

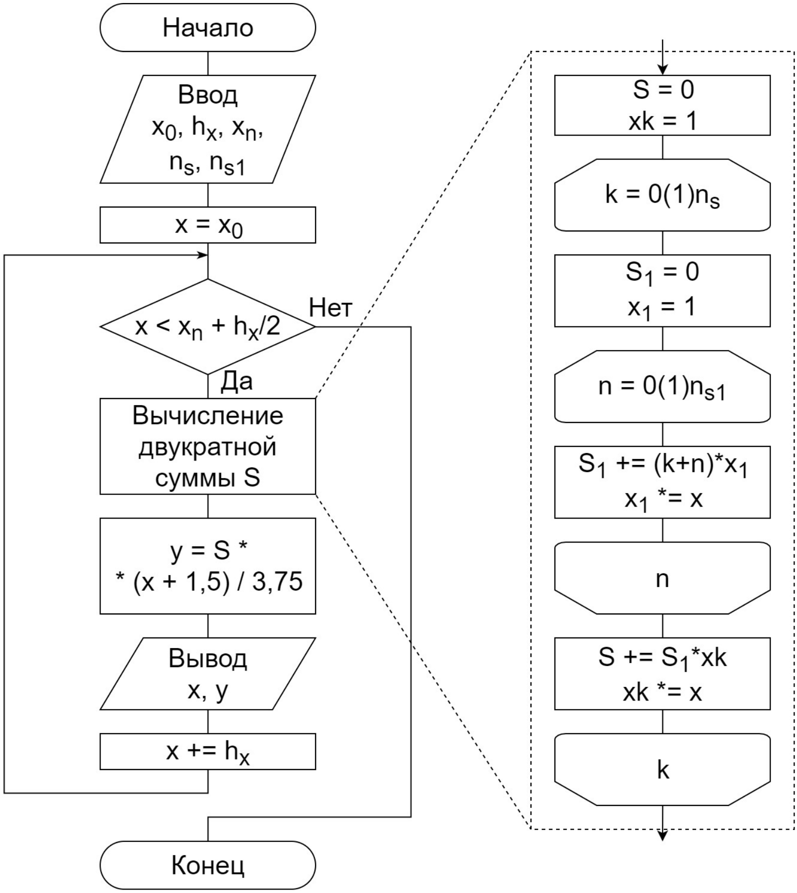


Рис. 3. Схема алгоритма вычисления кратной суммы

Составим программу:

```
# Цель: вычисление двойной суммы при различных
#        значениях аргумента x.
# Переменные:
#     s - сумма;
#     s1 - промежуточная сумма;
#     ns, ns1 - пределы сумм;
#     k, n - переменные суммирования;
#     xk - x в степени k;
```

```
#     x1 - x в степени n;
#     x = x0(hx)xn - параметры аргумента;
#     y - функция аргумента x.
# Программист: Степанов Н.Н.
# Дата написания: 12.04.2017.
print("Введите исходные данные:")

print("x0 = ", end='')
x0 = float(input())
print("hx = ", end='')
hx = float(input())
print("xn = ", end='')
xn = float(input())
print("ns = ", end='')
ns = int(input())
print("ns1 = ", end='')
ns1 = int(input())

print("Вы ввели:")
print("x0 = %.2f \t hx = %.2f \t xn = %.2f \t ns = %d
                 \t ns1 = %d" % (x0, hx, xn, ns, ns1))
# Табулирование функции
x = x0
while x < xn + hx / 2:
    # Вычисление двойной суммы
    s = 0
    xk = 1
    for k in range(0, ns):
        s1 = 0
        x1 = 1
        for n in range(0, ns1):
            s1 += (k + n) * x1
            x1 += x
        s += s1 * xk
        xk *= x
    y = (x + 1.5) * s / 3.75
    print("x = %.2f \t y = %.2f" % (x, y))
    x += hx
```

Нисходящее проектирование алгоритма и программы со структурой вложенных циклов

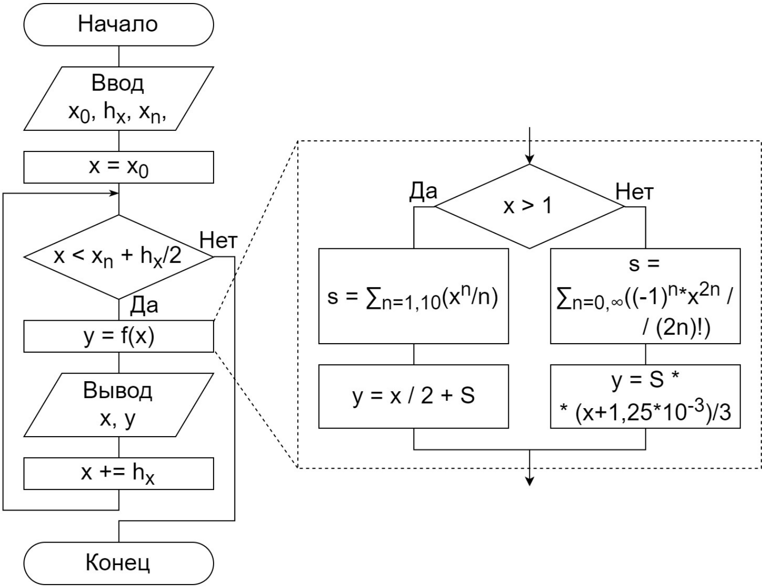
Процесс нисходящего проектирования рассмотрим на примере вычисления сложной функции

$$y = \begin{cases} \dfrac{x}{2} + \sum\limits_{n=1}^{10} \dfrac{x^n}{n}, & x > 1; \\ \dfrac{x + 1{,}25 * 10^{-3}}{3} \sum\limits_{n=0}^{\infty} (-1)^n \dfrac{x^{2n}}{(2n)!}, & x \leq 1 \end{cases}$$

при изменении аргумента по закону $x = x_0(h_x)x_n$. Вычисление бесконечной суммы необходимо реализовать с заданной погрешностью ε.

Анализ приведенного примера в общем виде свидетельствует о том, что требуется провести табулирование функции $y = f(x)$ при изменении аргумента по закону $x = x_0(h_x)x_n$. На рис. 4, а показан алгоритм табулирования функции, что иллюстрирует первый (самый общий) уровень детализации алгоритма решения задачи. Нетрудно заметить, что функция $y = f(x)$ является сложной и определяется по разным формулам в зависимости от выполнения / невыполнения условия $x > 1$. На рис. 4, б показан алгоритм вычисления сложной функции, что является следующим этапом детализации.

В процессе дельнейшей детализации необходимо уточнять алгоритмы вычисления конечной и бесконечной суммы. В разделе «Оператор цикла с заголовком. Вычисление конечных сумм и произведений» определены методы вычисления конечных сумм. Используя приведенные рекомендации, на рис. 5 приведена детализация блока вычисления конечной суммы. Используя рекомендации раздела «Итерационный цикл. Вычисление суммы бесконечного ряда» реализуется детализация блока вычисления суммы бесконечного ряда с заданной погрешностью ε (см. рис. 6).



а б

Рис. 4. Первые два уровня детализации алгоритма

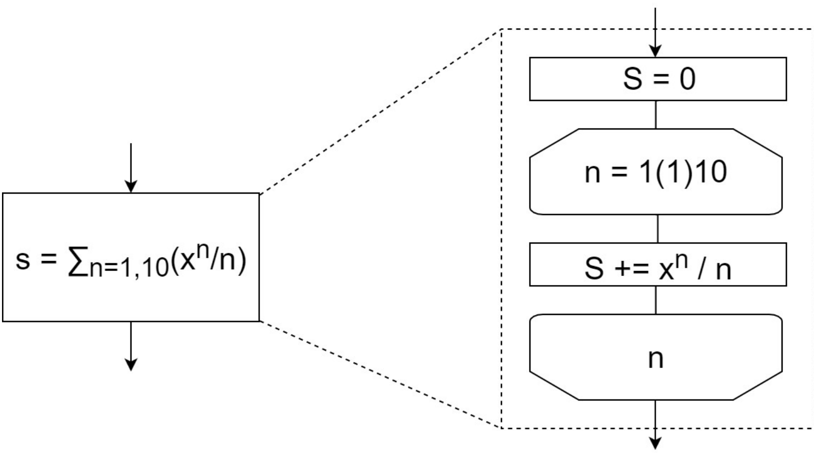


Рис. 5. Детализация блока вычисления конечной суммы

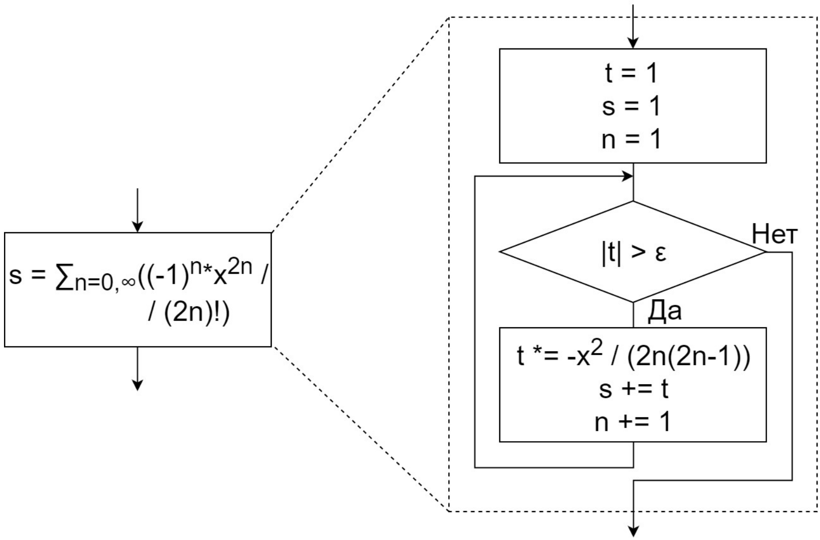


Рис. 6. Детализация блока вычисления бесконечной суммы

Проведенная пошаговая детализация позволяет достаточно просто представить общий алгоритм решения поставленной задачи. При этом общий алгоритм имеет структуры вложенных циклов. Соответствующая программа имеет следующий вид:

```
import math

# Цель: табулирование значений сложной функции,
#        зависящей от одной из сумм.
# Подпрограммы:
#     finalSum(x, n) - конечная сумма;
#     endlessSum(x, eps) - беск. сумма.
# Переменные:
#     s - сумма;
#     x = x0(hx)xn - параметры аргумента;
#     y - функция аргумента x.
# Программист: Степанов Н.Н.
# Дата написания: 12.04.2017.
```

```
def finalSum(x, n):
    s = 0
    for i in range(0, n):
        s += pow(x, n) / n
    return s

def endlessSum(x, eps):
    t = 1
    s = 1
    n = 1
    while math.fabs(t) > eps:
        t *= - math.pow(x, 2) / (2 * n * (2 * n - 1))
        s += t
        n += 1
    return s

print("Введите исходные данные:")

print("x0 = ", end='')
x0 = float(input())
print("hx = ", end='')
hx = float(input())
print("xn = ", end='')
xn = float(input())

print("Вы ввели:")
print("x0 = %.2f \t hx = %.2f \t xn = %.2f"
      % (x0, hx, xn))

x = x0
while x < xn + hx / 2:
    if x > 1:
        s = finalSum(x, 10)
        y = x / 2 + s
    else:
        s = endlessSum(x, 1e-3)
        y = s * (x + 1.25 * 1e-3 / 3)

    print("x = %.2f \t y = %.2f" % (x, y))
    x += hx
```

Контрольные вопросы

1. Какой алгоритм имеет структуру вложенных циклов?
2. Где располагается внутренний цикл в структуре вложенного цикла?
3. Что называется уровнем вложения цикла?
4. Как определить число повторений самого внутреннего цикла?
5. В чем сущность нисходящего поэтапного проектирования алгоритма со структурой вложенного цикла?
6. Какова структура алгоритма для табулирования функции трех переменных?
7. Как определить объем выводимых данных при табулировании функции трех переменных?
8. Приведите пример алгоритма со структурой вложенного цикла.
9. Какова структура алгоритма для нахождения кратной суммы?
10. В чем состоит идея автоматического выбора шага при численном интегрировании?

Задания

Используя метод нисходящего пошагового проектирования, разработать схему алгоритма и составить программу для вычисления функции при заданных значениях аргументов для своего варианта задания. Вывести результаты вычислений.

<u>Варианты заданий:</u>

1. *x = 0,1(0,1)0,9*;

$$y = \frac{\sin x + 2}{3 + \cos x}\sum_{n=0}^{20} a x^n; \quad a = \begin{cases} 2n, & x \le 0{,}5; \\ \dfrac{n}{2}, & x > 0{,}5. \end{cases}$$

2. *x = 0,2(0,2)1,8*;

$$y = \frac{a x^2}{\sqrt{x + a}} \quad a = \begin{cases} \displaystyle\sum_{k=1}^{12} \frac{2kx}{x + k^2}, & x < 1; \\ 1, & x \ge 1. \end{cases}$$

3. *x = 0,6(0,2)1,8*;

$$z = \sum_{k=0}^{10} \ln x \cdot \sin k\,(x - a); \quad a = \begin{cases} \dfrac{\pi}{4}, & x \le 1; \\ \pi, & x > 1. \end{cases}$$

4. *a = 0,5(0,2)1,9*; *x = 0,2(0,5)2,2*;

$$w = x^2\cos(ax+t)\,; \quad t = \begin{cases} 2ax, & a \ge x; \\ \dfrac{a}{2x}, & a < x. \end{cases}$$

5. *a = 2(0,5)8*;

$$z = \begin{cases} \sum\limits_{n=1}^{10} \dfrac{a^2}{a^n - 5}, & a < 4; \\ \dfrac{a+1}{a}\prod\limits_{n=1}^{8}\dfrac{a-1}{n}, & a \ge 4. \end{cases}$$

6. *x = 1(1)4*;

$$P = (xt)!; \quad n! = 1\cdot 2 \ldots n; \quad t = \begin{cases} 1.5, & x - \text{четное}; \\ 2, & \text{x} - \text{нечетное}. \end{cases}$$

7. *x = –0,5(0,1)0,5*;

$$z = \begin{cases} \ln(1-x)\,, & x \le 0; \\ \ln(1+x)\,, & x > 0. \end{cases}$$

Для вычисления *ln(1–x)* воспользоваться равенством

$$\ln(1-x) = -\sum_{n=1}^{50}\frac{x^n}{n}.$$

8. *a = 0,2(0,2)0,8*; *x = 1,1(0,5)2,6*;

$$F = \frac{a+x}{3}\sum_{n=0}^{6}(x+a)^{\frac{n}{2}}.$$

9. *a = 2*; *x = 1(0,5)3*;

$$z = \begin{cases} a\ln x\,, & x \ge a; \\ x\ln a\,, & x < a. \end{cases}$$

Для вычисления *ln(x)* воспользоваться равенством

$$\ln x \approx 2\sum_{n=0}^{10}\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}.$$

10. *x = 1,1(0,1)1,2*;

$$y = \prod_{n=1}^{8}\left(p - \frac{x^n}{2n+1}\right); \quad p = \begin{cases} 1, & n \le 5; \\ 2, & n > 5. \end{cases}$$

11. *x = 0,5(0,15)2*;

$$y = \sum_{k=1}^{10}(k + a\ln x); \quad a = \begin{cases} -1, & x \le 1; \\ 1, & x > 1. \end{cases}$$

12. *x = 1,28*;

$$s = \sum_{k=1}^{5}\sum_{n=1}^{10}(ax)^{\frac{k}{n}}; \quad a = \begin{cases} 0.1, & k \ge n; \\ 1, & k < n. \end{cases}$$

13. *a = 1(0,05)1,3*;

$$f = \sum_{n=0}^{5} (a+n)^n.$$

14. *a = 1,1(0,1)1,6*;

$$t = a \sum_{i=1}^{10} \sum_{j=1}^{i} \frac{i}{j + a^j}.$$

15. *a = 1,45*; *x = 1(0,1)1,8*;

$$y = \begin{cases} \sum_{k=1}^{10} a^k x^k, & a \le x; \\ \prod_{k=1}^{8} (a^k - x^k), & a > x. \end{cases}$$

16. *x = 0,5(0,5)4*;

$$y = \begin{cases} \prod_{n=1}^{8} \left(\frac{x}{n}\right)^n, & x \le 2; \\ \sum_{n=0}^{5} (1 + xn), & x > 2. \end{cases}$$

17. *x = 0(1)6*;

$$f = e^{-x} - x!; \quad x! = \prod_{n=1}^{x} n; \quad 0! = 1.$$

18. $x = -\pi\left(\frac{\pi}{4}\right)\pi$;

$$y = \prod_{n=1}^{5} \left(\frac{1}{n} + a \sin x\right); \quad a = \begin{cases} 1, & x \ge 0; \\ -1, & x < 0. \end{cases}$$

19. *x = –0,5(1)4,5*;

$$y = \prod_{k=0}^{8} \prod_{n=1}^{5} \left(k + \frac{\sin x}{n}\right).$$

20. *x = 1(0,05)1,2*;

$$y = \sum_{k=0}^{6} \sum_{n=0}^{4} (k+n) x^{k+n}.$$

Литература

1. Python на примерах. Практический курс по программированию. / А.Н. Васильев. – СПб.: Наука и Техника, 2016. – 432 с.
2. http://pythonicway.com
3. https://pythonworld.ru
4. http://www.python-course.eu/python3_course.php

Содержание