**Лабораторная работа №3**

**Тема: Детерминированные вычислительные процессы с управлением по аргументу. Численное интегрирование**

**Цель: Научиться реализовывать алгоритм детерминированных вычислительных процессов средством при Pascal**

**Оборудование:**

**• ПК**

**• Среда программирования Lazarus**

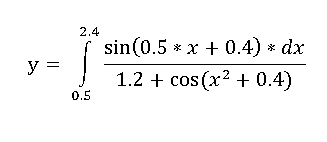
**• Интернет**

**Задача 1**

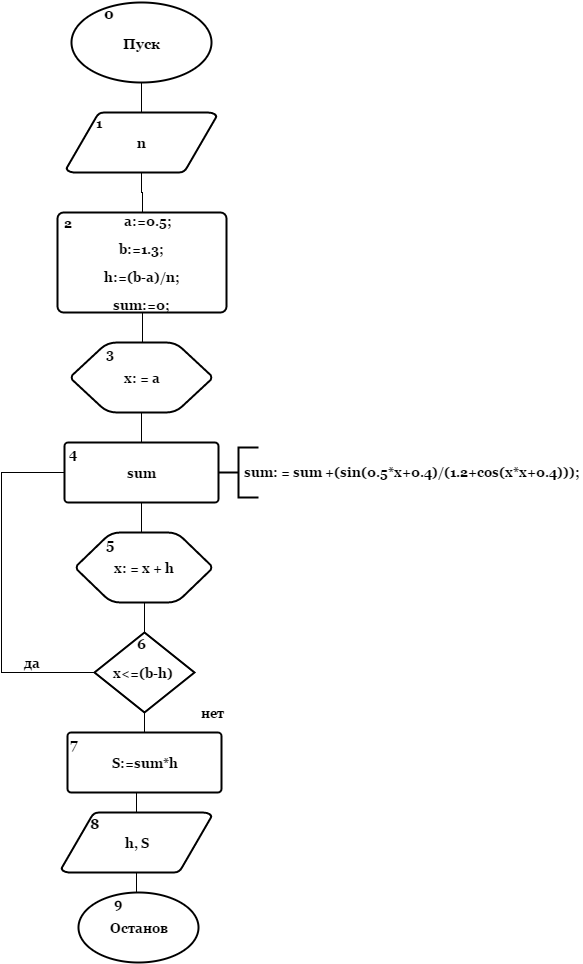
**Постановка задачи:**

Написать программу для вычисления определенного интеграла из индивидуального задания *методом прямоугольника левых частей*. Протестировать программу на определенном интеграле, вычисленном в ходе выполнения самостоятельной работы 3

**Математическая модель:**



**Блок – схема:**



**Список идентификаторов:**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Смысл** |
| a | real | начало отрезка |
| b | real | конец отрезка |
| n | integer | количество разбиений |
| h | real | шаг |
| S | real | значение интеграла |
| x | real | аргумент |
| sum | real | сумма функций |

**Программный код:**

program lev;

var

n:integer;

a,b,h,x,S,sum:real;

begin

Write('Vvedite n - ');

readln (n);

a:=0.5;

b:=1.3;

h:=(b-a)/n;

sum:=0;

x:=a;

writeln('h = ',h:3:5);

while x<=(b-h) do

begin

sum:=sum+(sin(0.5\*x+0.4)/(1.2+cos(x\*x+0.4)));

x:=x+h;

end;

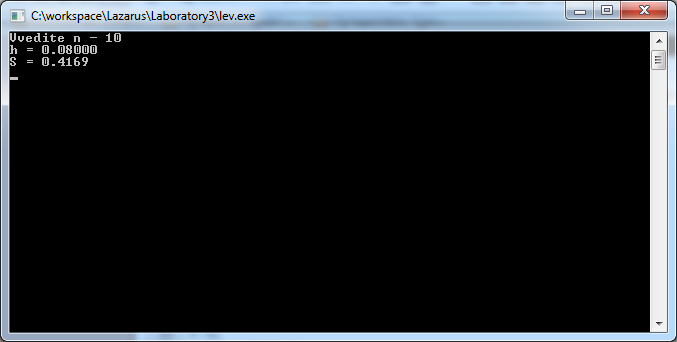
S:=sum\*h;

Writeln('S = ',S:3:4);

readln;

end.

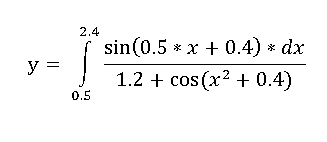
**Результат выполнения программы:**



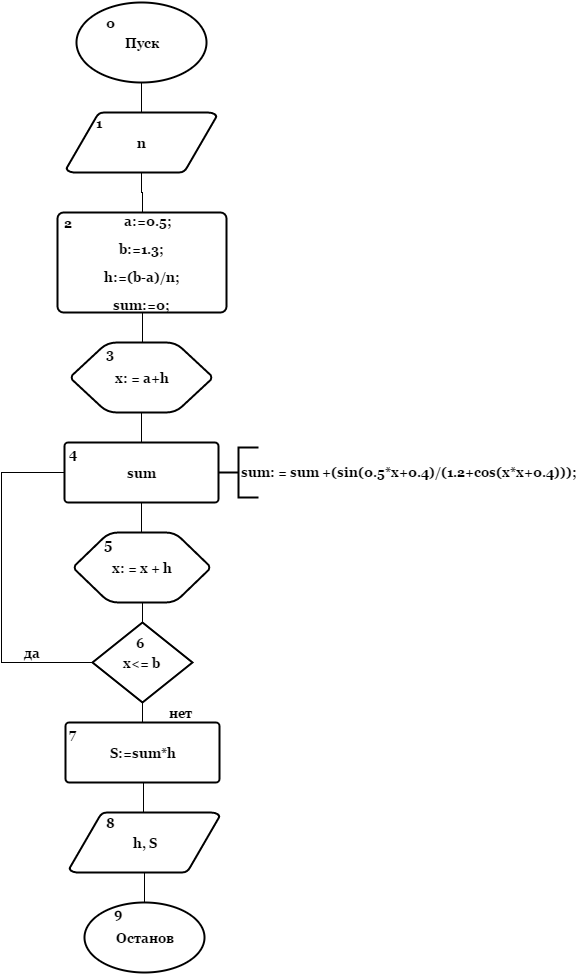
**Задача 2**

**Постановка задачи:** Написать программу для вычисления определенного интеграла из индивидуального задания *методом прямоугольника правых частей*. Протестировать программу на определенном интеграле, вычисленном в ходе выполнения самостоятельной работы 3

**Математическая модель:**



**Блок – схема:**



**Список идентификаторов:**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Смысл** |
| a | real | начало отрезка |
| b | real | конец отрезка |
| n | integer | количество разбиений |
| h | real | шаг |
| S | real | значение интеграла |
| x | real | аргумент |
| sum | real | сумма функций |

**Программный код:**

program prav;

var

n:integer;

a,b,h,x,S,sum:real;

begin

Write('Vvedite n - ');

readln (n);

a:=0.5;

b:=1.3;

h:=(b-a)/n;

sum:=0;

x:=a+h;

writeln('h = ',h:3:5);

while x<=b do

begin

sum:=sum+(sin(0.5\*x+0.4)/(1.2+cos(x\*x+0.4)));

x:=x+h;

end;

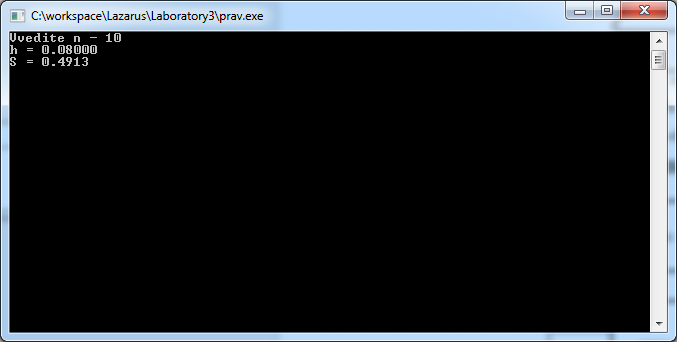
S:=sum\*h;

Writeln('S = ',S:3:4);

readln;

end.

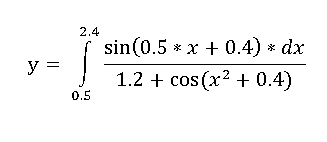
**Результат выполнения программы:**



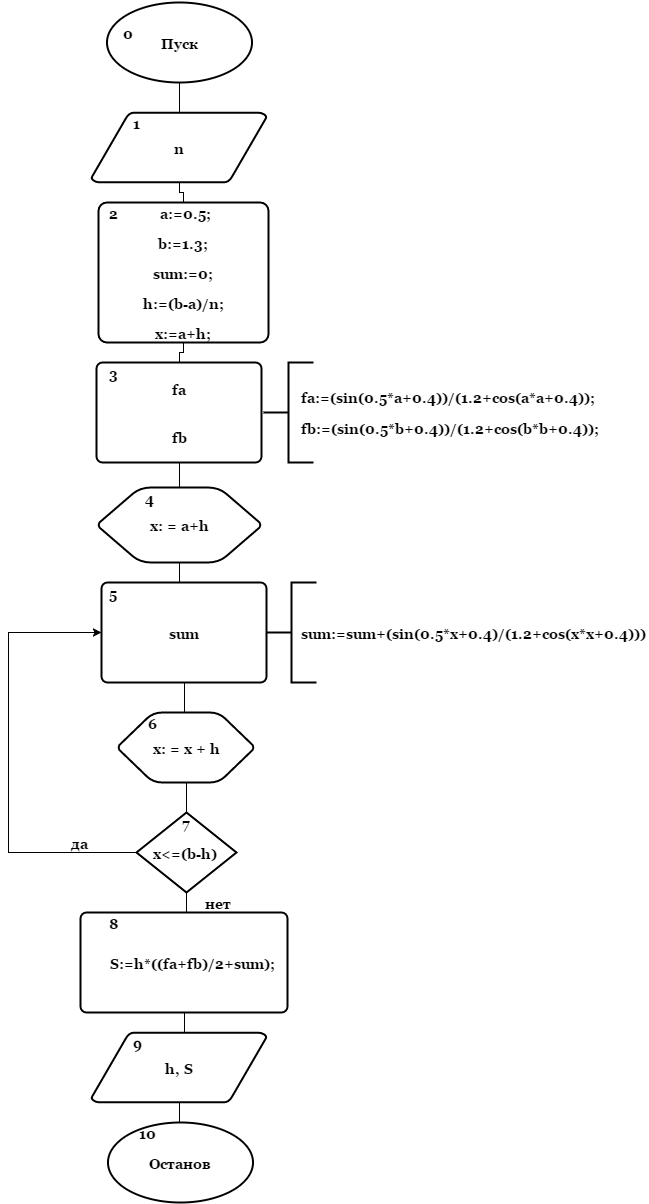
**Задача 3**

**Постановка задачи:** Написать программу для вычисления определенного интеграла из индивидуального задания *методом трапеций*. Протестировать программу на определенном интеграле, вычисленном в ходе выполнения самостоятельной работы 3

**Математическая модель:**



**Блок – схема:**



**Список идентификаторов:**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Смысл** |
| a | real | начало отрезка |
| b | real | конец отрезка |
| n | integer | количество разбиений |
| h | real | шаг |
| S | real | значение интеграла |
| x | real | аргумент |
| sum | real | сумма функций |
| fa | real | начальное значение функции |
| fb | real | конечное значение функции |

**Программный код:**

program trapec;

var

n:integer;

a,b,h,x,S,fa,fb,sum:real;

begin

Write('Vvedite n - ');

readln (n);

a:=0.5;

b:=1.3;

sum:=0;

h:=(b-a)/n;

x:=a+h;

fa:=(sin(0.5\*a+0.4))/(1.2+cos(a\*a+0.4));

fb:=(sin(0.5\*b+0.4))/(1.2+cos(b\*b+0.4));

writeln('h = ',h:3:5);

while x<=(b-h) do

begin

sum:=sum+(sin(0.5\*x+0.4)/(1.2+cos(x\*x+0.4)));

x:=x+h;

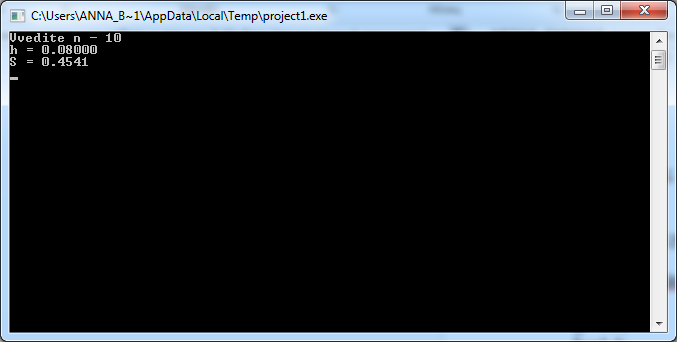
end;

S:=h\*((fa+fb)/2+sum);

Writeln('S = ',S:3:4);

readln;

end.

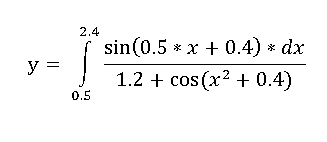
**Результат выполнения программы:**

**Задача 4**

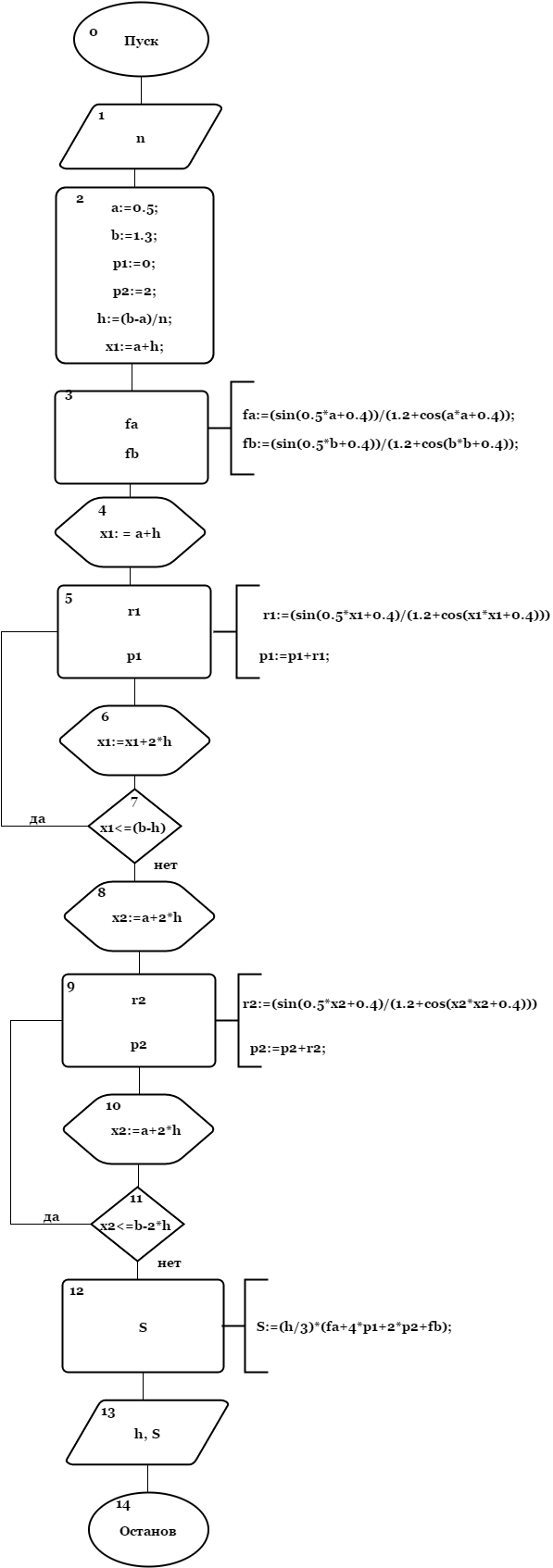
**Постановка задачи:**

Написать программу для вычисления определенного интеграла из индивидуального задания *методом парабол*. Протестировать программу на определенном интеграле, вычисленном в ходе выполнения самостоятельной работы 3

**Математическая модель:**



**Блок – схема:**



**Список идентификаторов:**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Смысл** |
| a | real | начало отрезка |
| b | real | конец отрезка |
| n | integer | количество разбиений |
| h | real | шаг |
| S | real | значение интеграла |
| x1 | real | аргумент для нечётных |
| x2 | real | аргумент для чётных |
| r1 | real | значение нечётной функции |
| r2 | real | значение чётной функции |
| p1 | real | сумма нечётных функций |
| p2 | real | сумма чётных функций |
| fa | real | начальное значение функции |
| fb | real | конечное значение функции |

**Программный код:**

program parabol;

var

n:integer;

a,b,h,x1,x2,r1,r2,p1,p2,S,fa,fb:real;

begin

Write('Vvedite n - ');

readln (n);

a:=0.5;

b:=1.3;

p1:=0;

p2:=0;

h:=(b-a)/n;

x1:=a+h;

fa:=(sin(0.5\*a+0.4))/(1.2+cos(a\*a+0.4));

fb:=(sin(0.5\*b+0.4))/(1.2+cos(b\*b+0.4));

writeln('h = ',h:3:5);

while x1<=(b-h) do

begin

r1:=(sin(0.5\*x1+0.4)/(1.2+cos(x1\*x1+0.4)));

p1:=p1+r1;

x1:=x1+2\*h;

end;

x2:=a+2\*h;

while x2<=b-2\*h do

begin

r2:=(sin(0.5\*x2+0.4)/(1.2+cos(x2\*x2+0.4)));

p2:=p2+r2;

x2:=x2+2\*h;

end;

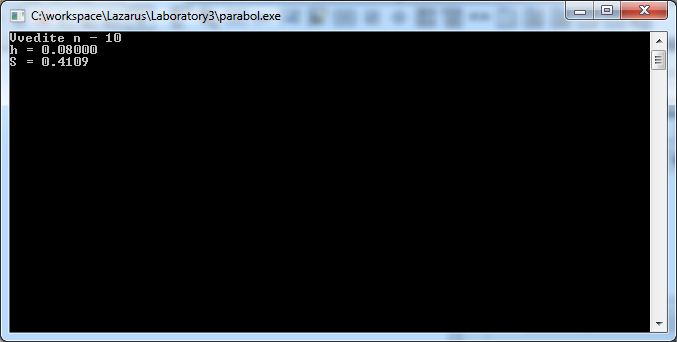
S:=(h/3)\*(fa+(4\*p1)+(2\*p2)+fb);

Writeln('S = ',S:4:4);

readln;

end.

**Результат выполнения программы:**



**Результат:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N Количество разбиений | H шаг | I Метод левых частей прямоугольника | I Метод правых частей прямоугольника | I Метод трапеций | I Метод парабол |
| 10 | 0,08 | 0,4169 | 0,4913 | 0,4541 | 0,4109 |
| 100 | 0,008 | 0,4387 | 0,4458 | 0,4424 | 0,4329 |
| 1000 | 0,0008 | 0,4516 | 0,4523 | 0,4519 | 0,4500 |
| 10000 | 0,00008 | 0,4518 | 0,4519 | 0,4519 | 0,4518 |

**Вывод:**

1. Самый точный метод - метод левых частей прямоугольника, так как разница очень небольшая между разными количествами шагов, по сравнению с остальными методами.
2. Для уменьшения погрешности отрезок интегрирования разбивают на части и применяют численный метод для оценки интеграла на каждой из них. При стремлении количества разбиений к бесконечности, оценка интеграла стремится к его истинному значению для аналитических функций для любого численного метода. Приведённые выше методы допускают простую процедуру уменьшения шага в два раза, при этом на каждом шаге требуется вычислять значения функции только во вновь добавленных узлах.