**Семинар 4. Особенности обработки итерационных циклических процессов. Численные методы.**

**Циклическими называются процессы, при которых некоторая последовательность действий повторяется более одного раза.**

**В зависимости от способа организации самого процесса и выхода из него, циклы делятся на**

**- Счетные.**

**- Итерационные.**

**-Поисковые. (см. презентацию 2).**

**Итерационные циклы.**

Итерационными называются циклы с заранее неизвестным числом повторений. Они характеризуются постепенным приближением к требуемому значению с заданной точностью или достижением некоторого другого условия.

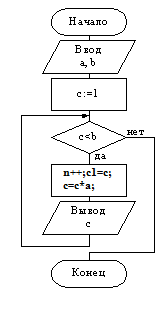
**Пример 1**. Задано действительное число *a>1*. Получить все члены последовательности *a*, *a*2, *a*3, …, не превышающие действительного числа *b*.

Из условия задачи ясно, что исходными данными являются числа *a* и *b*, которые, например, вводятся с клавиатуры. Именно с их помощью будет формироваться нужное число степеней *a*. Степени можно получить последовательным умножением *a* на само себя. Результат умножения будем хранить во вспомогательной переменной *c*. Эти действия и являются телом цикла. Так как *a0=1*, то в блоке подготовки цикла переменной *c* устанавливаем начальное значение, равное 1. Условие выхода из цикла – это превышение числа *b* числом *c*.

Однако, нам нужно найти с не превышающие действительного числа *b*.

Так как выход из цикла происходит по c>=b, то нам нужно не текущее, а предыдущее c. Для этого необходима переменная с1, которая хранит предыдущее значение an. n- номер итерации и очередная степень a.

Схема алгоритма решения задачи представлена на рисунке.

****

**Текст программы**

**// Найти все члены последовательности a^1, a^2, a^3, …, которые < b (a>1)**

**void main(int argc, char\* argv[])**

**{ float c=1,c1,a, b;int n=0;**

**puts("Input a,b >1");**

**scanf("%f %f",&a,&b);**

**printf("a=%6.2f b=%7.3f\n",a,b);**

**puts(" n a c");**

**while (c<b)**

**{ n++;c1=c;**

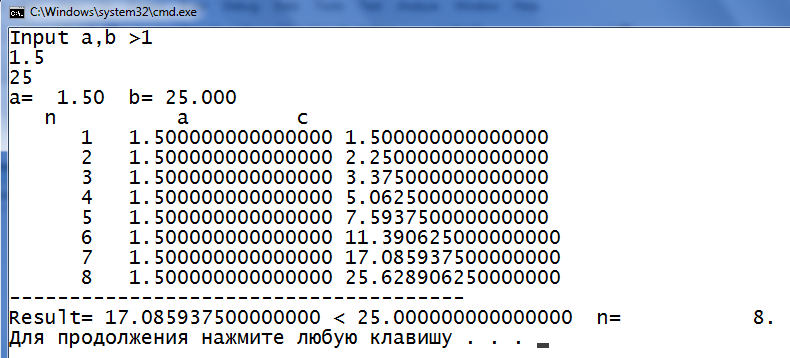
**c=c\*a;**

**printf(" %4d %16.15f %16.15f\n", n,a,c);**

**}**

**printf("--------------------------------------\nResult= %16.15f < %16.15f n=%12d.\n", c1,b,n);**

**}**

**Пример работы программы**

**Пример 2.** **Найти сумму ряда**

**y=sin(x)=x-x^3/3!+x^5/5!......**

**с заданной точностью e.**

Как известно из математики, почти все тригонометрические функции считаются приближенными методами с помощью рядов. Попробуем решить эту задачу численными методами. Для этого необходимо знать либо формулу общего члена ряда, либо определить закономерности расчета следующего члена через предыдущий. Если известна формула общего члена, то нужно следующий член ряда поделить на предыдущий и найти сомножитель для расчета.

**<добавка> =Rn/Rn-1=Rn+1/Rn;**

Так как нам неизвестна формула общего члена ряда, то пойдем путем рассуждений:

R1=x^1/1!; 1!=1

R2=-x^3/3!; 3!=1\*2\*3=1!\*2\*3

R3=x^5/5!; 5!=1\*2\*3\*4\*5=3!\*4\*5

…..

Rn=±x^n/((2\*n-1)!= (2\*n-1)!=n!\*(2\*n)\*(2\*n+1)

Как не сложно видеть, чтобы из предыдущего члена получить следующий необходимо:

Rn=-Rn-1\*x^2/((2\*n)\*(2\*n+1))

Так как член ряда меняет знак, то искать член ряда меньший точности **е** нужно по абсолютной величине.

**Текст программы**

**#include <stdio.h>**

**#include <math.h>**

**// Найти сумму ряда y=sin(x)=x-x^3/3!+x^5/5!......**

**void main()**

**{ float s,x, r,eps;int i,n=0;**

**puts("Input x,eps");**

**scanf("%f %f",&x,&eps);**

**printf("x=%6.2f eps=%9.8f\n",x,eps);**

**s=x; r=x;i=0;**

**while (fabs(r)>eps) // условие нахождения суммы с точностью eps.**

**{ i++;**

**r=-r\*x\*x/(2\*i)/(2\*i+1); // текущий член ряда – рекуррентное соотношение**

**s=s+r;**

**n++;**

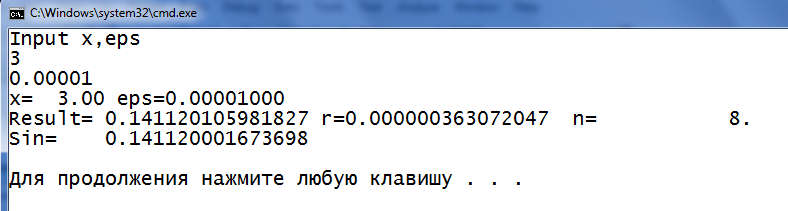
**}**

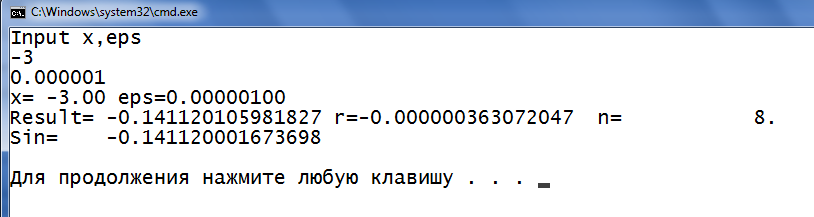
**printf("Result= %16.15f r=%16.15f n=%12d.\n", s,r,n);**

**// проверка с помощью стандартной функции.**

**printf("Sin= %16.15f\n\n", sin(x)); }**

Примеры работы программы





**Пример 3. Найти сумму ряда**

**arcsin(x)= x+(1/2\*x^3/3+1/2\*3/4\*x^5/5)+(1/2\*3/4\*5/6\*x^7/7)+....**....

Так как арксинус, это угол, синус которого равен x, то **0<=|x|<=1**. И это нужно контролировать при вводе.

Формулы общего члена ряда нет, поэтому, путем рассуждений, как в предыдущем примере можно найти зависмость следующего члена ряда через предыдущий

**Rn=Rn-1\*x^2\*(2\*n-1)^2/((2\*n)(2\*n+1))**

Текст программы

**#include <stdio.h>**

**#include <math.h>**

**// найти сумму ряда arcsin(x)=**

**// x+(1/2\*x^3/3+1/2\*3/4\*x^5/5)+(1/2\*3/4\*5/6\*x^7/7)+........**

**void main()**

**{ float s,x, r,eps;int i,n=0;**

**puts("Input x,|x|<=1,eps");**

**scanf("%f %f",&x,&eps);**

**printf("x=%6.2f eps=%9.8f\n",x,eps);**

**s=x; r=x;i=0;**

**while (fabs(r)>eps)**

**{ i++;**

**r=r\*x\*x\*(2\*i-1)\*(2\*i-1)/(2\*i)/(2\*i+1); // текущий член ряда – рекуррентное соотношение**

**s=s+r;**

**n++;**

**}**

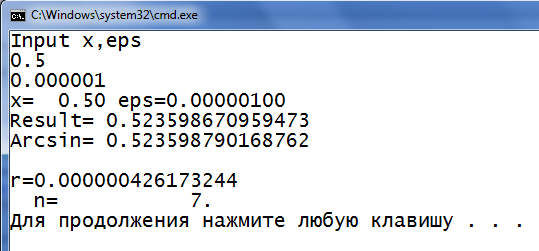
**printf("Result= %16.15f\n", s);**

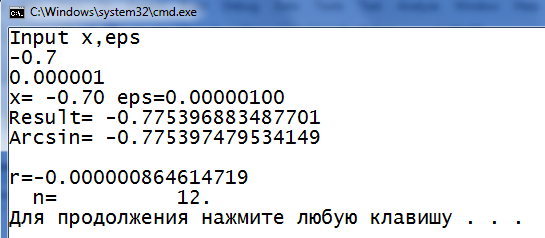
**printf("Arcsin= %16.15f\n\n", asin(x)); // Точное значение по функции**

**printf("r=%16.15f\n n=%12d.\n", r,n);**

**}**

**Примеры работы программы**

****



В конкретных случаях циклических процессов указанные выше этапы цикла могут присутствовать в различных сочетаниях, однако порядок их проектирования не меняется:

1. определяются действия, составляющие тело цикла;
2. заполняется блок подготовки итерации;
3. определяется условие выхода из цикла (счетчик или логическое выражение);
4. заполняется блок подготовки цикла.

Для решения некоторых разновидностей задач необходимо использовать вложенные циклы.

**Вложенными** циклическими процессами называются такие процессы, при которых внутри одного циклического процесса, происходит другой.

Каждый из процессов может реализоваться различными операторами цикла.

Внешний цикл может быть счетным, а внутренний – итерационным и наоборот.

На количество вложенных циклов компилятор С++ не накладывает никаких ограничений. Оно определяется логикой программы и желанием программиста.

При программировании циклов необходимо соблюдать правило строгой вложенности – начала и концы циклов не должны перекрещиваться, а каждый вложенный цикл иметь начало и конец внутри внешнего цикла.

Вход внутрь цикла по goto возможен только через его начало.

Пример 4.Вычислить определенный интеграл функции f(x) на интервале [a,b] методом прямоугольников с точностью d.

Итак n



S = f(x1)\*d + f(x2)\*d + f(x3)\*d+ …+ f(xn)\*d = d\*∑f(xi), где d=(b-a)/n. i=1

Увеличивая **n**, получаем приближения площади: **S1, S2, S3 ...**

Останавливаемся, когда |Sk-Sk+1| < d

**Обобщенный алгоритм:**

*Шаг 1.* Ввести a, b, d.

*Шаг 2*. Задать число прямоугольников n:=10.

*Шаг 3*. Определить шаг d:=(b-a)/n.

*Шаг 4*. Определить площадь фигуры S1.

*Шаг 5*. Увеличить число прямоугольников вдвое n:=n\*2.

*Шаг 6*. Уменьшить шаг вдвое d:=d/2.

*Шаг 7*. Определить площадь фигуры S2. **// внутренний цикл**

*Шаг 8*. Если Разность площадей меньше d, то перейти к шагу 11

*Шаг 9*. Запомнить новое значение площади S1:=S2.

*Шаг 10.* Перейти к шагу 5.

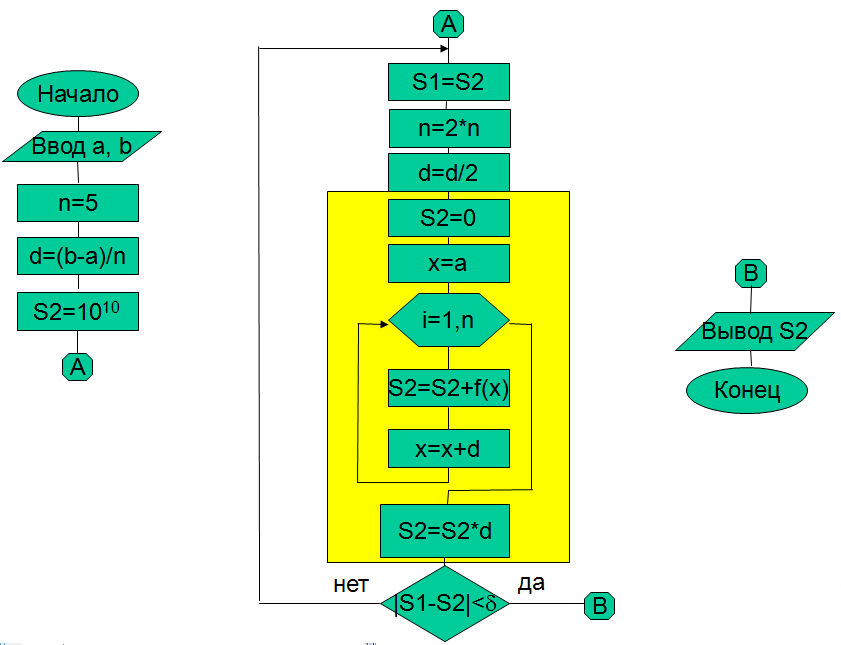
*Шаг 11.* Вывести S2.

Конец.

Если мы нарисуем схему алгоритма, она получится не структурной (см. презентацию 2).

Для приведения алгоритма к структурному есть прием. Первая сумма берется очень большой, чтобы войти в цикл, а далее следуем алгоритму.

**Схема алгоритма**



**Текст программы**

**// Вычислить определенный интеграл функции f(x)=x^2-1; на интервале [a,b]**

**//методом прямоугольников с точностью eps**

**int main(int argc, char\* argv[])**

**{int i,n,k;**

**double s1,s2,x,a,b,eps,d;**

**puts("input a,b,eps");**

**scanf("%lf %lf %lf",&a,&b,&eps);**

**n=5;**

**d=(b-a)/n;**

**s2=1.0e+10;**

**k=0;**

**do //цик расчета разности площадей**

**{s1=s2;**

**s2=0;n=n\*2;**

**d=d/2;**

**x=a;k++;**

**for(i=1;i<=n;i++) //цик расчета очередной площади**

**{ s2=s2+x\*x-1; // f(x)=x^2-1;**

**x=x+d;**

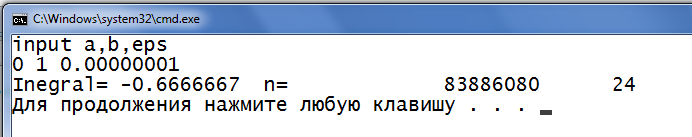
**}**

**s2=s2\*d;**

**} while(fabs(s2-s1)>eps);**

**printf("Inegral= %10.7lf n= %20d %6d\n",s2,n,k);**

**return 0;**

**}**

**Пример работы программы**

**Дома решить задачу из третьей части ДЗ1**