Итерационные циклы

Составляющие цикла:
1. начальное состояние;
2. ограничение цикла (условие продолжения или завершения):
3. переход к следующему шагу:
4. повторяющееся действие – тело цикла

«Хороший» цикл – части независимы:
- логический анализ
- количество шагов

						k=i
1	3	4	7	9	11	6
1	3	4	7	9	6	11
1	3	4	7	6	9	11
1	3	4	6	7	9	11

```
void sort(int in[],int n) {
for ( int i=1; i<n; i++)

        for ( int k=i; k !=0 && in[k] < in[k-1]; k--){
 int c=in[k]; in[k]=in[k-1]; in[k-1]=c;
 }}
```

Пример «плохого» цикла – в цикле перестановка элементов, ограничение – зависит от переставляемых (замкнутый круг, индуктивный подход к анализу)

Итерационные циклы

Итерационные циклы: состояние цикла на текущем шаге зависит от состояния на предыдущих шагах:

- количество шагов ???
- возможное зацикливание

```
int FIBO(int n){
        int Fn,Fn1=1,Fn2=1;
        for (int i=2; i<=n; i++){
            Fn=Fn1+Fn2;
            Fn2=Fn1;      // Переход к следующему шагу:
            Fn1=Fn;       // текущий становится предыдущим
                }
        return Fn;
        }
```

Рекуррентные последовательности:
Fn = Fn-1 + Fn-2, F0=1 F1=1

Fn = 2*Fn/3 + 3*Fn/2, F0=1 F1=1

```
int F21(int n){
    if (n<=1) return 1;
    return 3*F21(n/2)+2*F21(n/3);
    }
int F22(int n){
    int a[1000]={1,1};
    for(int i=2;i<=n;i++)
        a[i] = 3*a[i/2]+2*a[i/3];
    return a[n];
    }
```

Итерационные циклы

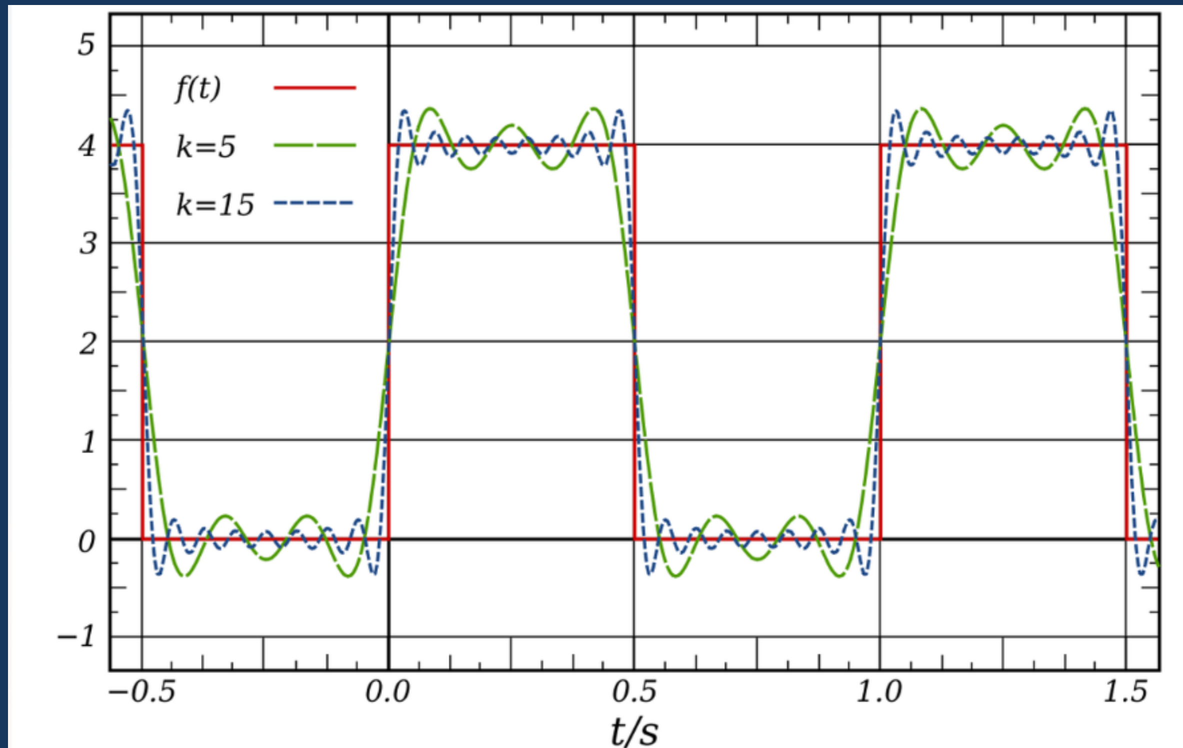
Вычисление sin/cos кратных углов

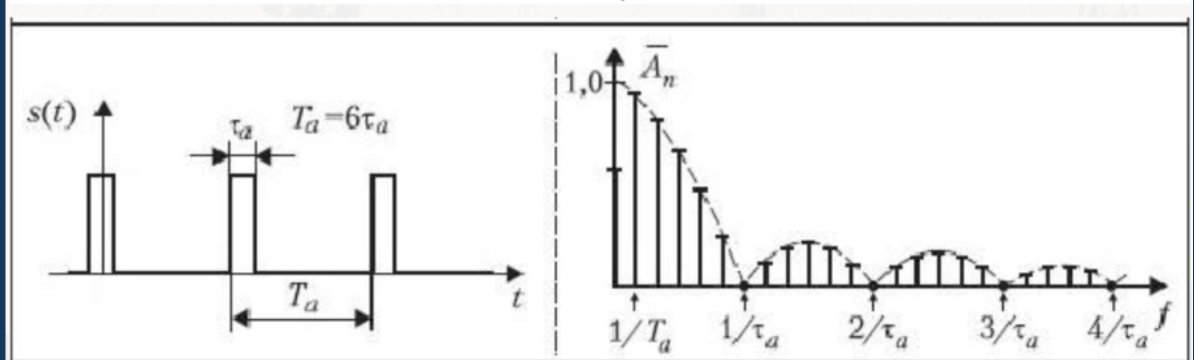
```
sin(nx) = sin((n-1)x)cos(x) + cos((n-1)x)sin(x)
cos(nx) = cos((n-1)x)cos(x) - sin((n-1)x)sin(x)
```

```c
#include <math.h>
void FSIN(){
int FI=30;
double pi=3.1415926;
double Cn,Cn1,C0,Sn,Sn1,S0;     // S0=sin(x), Sn1=sin((n-1)x), Sn=sin(nx)
Cn1=C0=cos(FI*pi/180);          // при n=2  sin((n-1)x)=sin(x)
Sn1=S0=sin(FI*pi/180);
printf("sin(%d*%d)=%lf cos(%d*%d)=%lf\n",1,FI,S0,1,FI,C0);
for (int n=2;n<=10;n++){
       Sn=Sn1*C0 + Cn1*S0;      // Рекуррентная формула
       Cn=Cn1*C0 - Sn1*S0;
       printf("sin(%d*%d)=%lf cos(%d*%d)=%lf\n",n,FI,Sn,n,FI,Cn);
       Cn1=Cn;                  // Переход к следующему шагу
       Sn1=Sn;                  // текущий становится предыдущим
       }
}
```

Итерационные циклы

Спектр периодического сигнала = сумма синусоид с кратными периодами (гармоники)



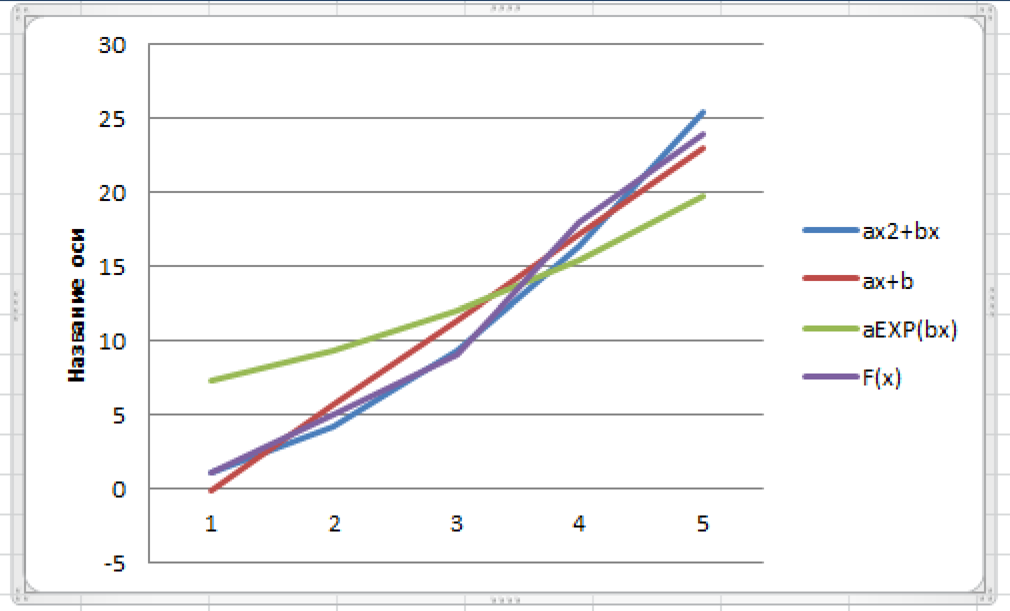


Итерационные циклы

Последовательное приближение – градиентный спуск:

- Имеется ряд значений F(x) для набора xi
- Оценить максимальное приближение (аппроксимация) функцией
  - Z(a,b,x) = ax+b
  - Z(a,b,x) = ax^2+bx
  - Z(a,b,x) = a e^{bx}
- найти a,b, при которых минимальная степень отклонения
- $\sqrt{(\sum(F(x) - Z(a,b,x))^2 / n)}$
- движение в сторону максимального уменьшения ОЦЕНКИ ОТКЛОНЕНИЯ
- a ± d, b ± d (4 соседние точки в пространстве [a,b])

	diff	1,05	1,35	4,28
	ns	22	218	128
	a	1	5,8	5,65
	b	0,1	-6	0,25
x	y	ax2+bx	ax+b	aEXP(bx)
1	1	1,1	-0,2	7,25
2	5	4,2	5,6	9,32
3	9	9,3	11,4	11,96
4	18	16,4	17,2	15,36
5	24	25,5	23	19,72



Итерационные циклы

```
double Fax2bx(double a, double b, double x){
    return a*x*x+b*x;
    }
double Faxb(double a, double b, double x){
    return a*x+b;
    }
double FaEXPbx(double a, double b, double x){
    return a*exp(b*x);
    }
double diff(double &a, double &b, double x[], double y[],int sz,
        double(*pf)(double,double,double)){
    double sum=0;
    for(int i=0;i<sz;i++){
        double v0 = (*pf)(a,b,x[i]) - y[i];
        sum += v0*v0;
        }
    return sqrt(sum/sz);
    }
```

```
ns=22 a=1.000000 b=0.100000, diff=1.053565
ns=218 a=5.800000 b=-6.000000, diff=1.356466
ns=128 a=5.650000 b=0.250000, diff=4.284962
```

Итерационные циклы

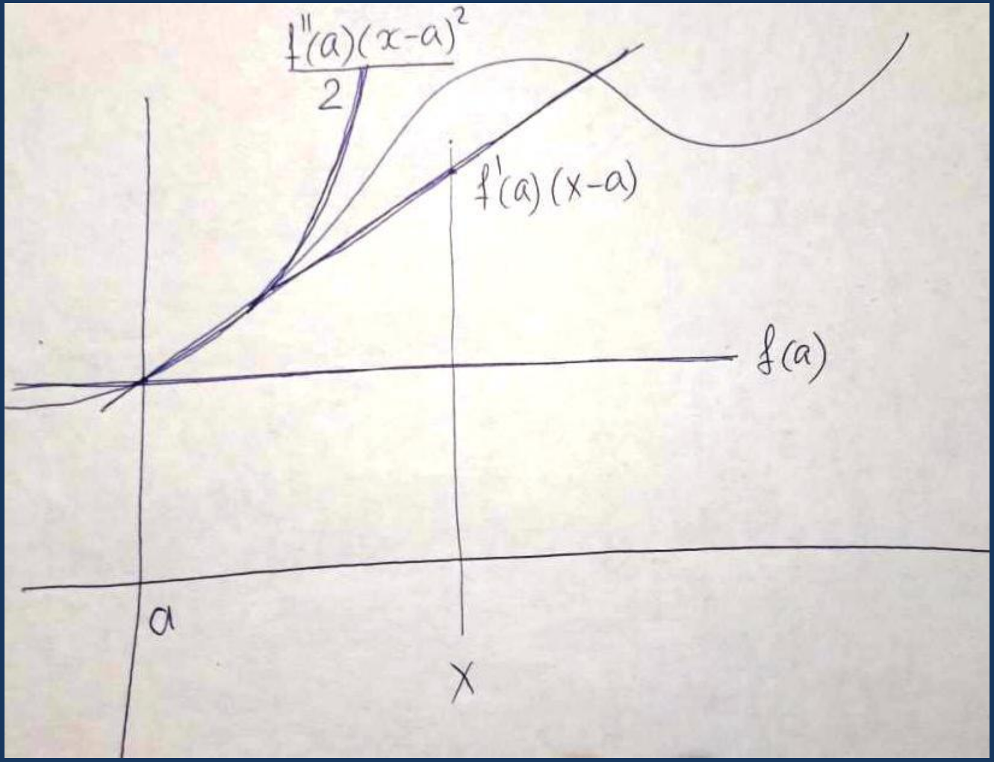
```
double iter(int &n,double dab,double &a, double &b, double x[], double y[],
        int sz,double(*pf)(double,double,double)){
    int ns=10000;
    double aa,bb, a1,b1,v0=0,v1;
    int found = 0;
    for(n=0;n<ns;n++){
        found = 0;
        v0 = diff(a,b,x,y,sz,pf);              // Текущая точка
        aa=a+dab; bb=b;
        v1 = diff(aa,bb,x,y,sz,pf);            // 4 соседа - fix уменьшения
        if (v1<v0){ v0 = v1; found=1; a1=aa; b1=bb; }
        aa=a-dab; bb=b;
        v1 = diff(aa,bb,x,y,sz,pf);
        if (v1<v0){ v0 = v1; found=1; a1=aa; b1=bb; }
        aa=a; bb=b+dab;
        v1 = diff(aa,bb,x,y,sz,pf);
        if (v1<v0){ v0 = v1; found=1; a1=aa; b1=bb; }
        aa=a; bb=b-dab;
        v1 = diff(aa,bb,x,y,sz,pf);
        if (v1<v0){ v0 = v1; found=1; a1=aa; b1=bb; }
        if (!found)
            return v0;                         // Нет уменьшения ОЦЕНКИ РАЗНИЦЫ
        else{
            a = a1; b = b1;                    // есть - СПУСК С ГОРЫ
            //printf("diff=%lf a=%lf b=%lf\n",v1,a,b);
            }
        }
    return v0;
    }
```

Вычисление степенного ряда

Степенной ряд – приближенное представление функции в окрестности точки а в виде суммы степеней отклонения x-a. Ряд Тейлора.

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k$$

$$f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\ldots+\frac{f^{(n)}(a)}{n!}(x-a)^n+$$



Вычисление степенного ряда

Индуктивный подход:

- последовательность коэффициентов перехода
- обнаружение закономерности
- запись в виде функции от номера шага

10	$x + x^3*1/(2*3) + x^5*(1*3)/(2*4*5) + \ldots$ $+x^{2n+1}(1*3*\ldots*2n-1)/(2*4*\ldots 2n(2n+1))$	arcsin(x)

n=0 S(n) = x

n=1 S(n) = x^3 *1 / 2* 3 k= x^2 * 1 / (2*3)

n=2 S(n) = x^5 *(1*3) / (2*4*5) k = x^2 * 3 * **3** / (4*5)

n=3 S(n) = x^7 *(1*3*5) / (2*4*6*7) k = x^2 * 5 * **5** / (6*7)

второй множитель сокращает предыдущий знаменатель

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

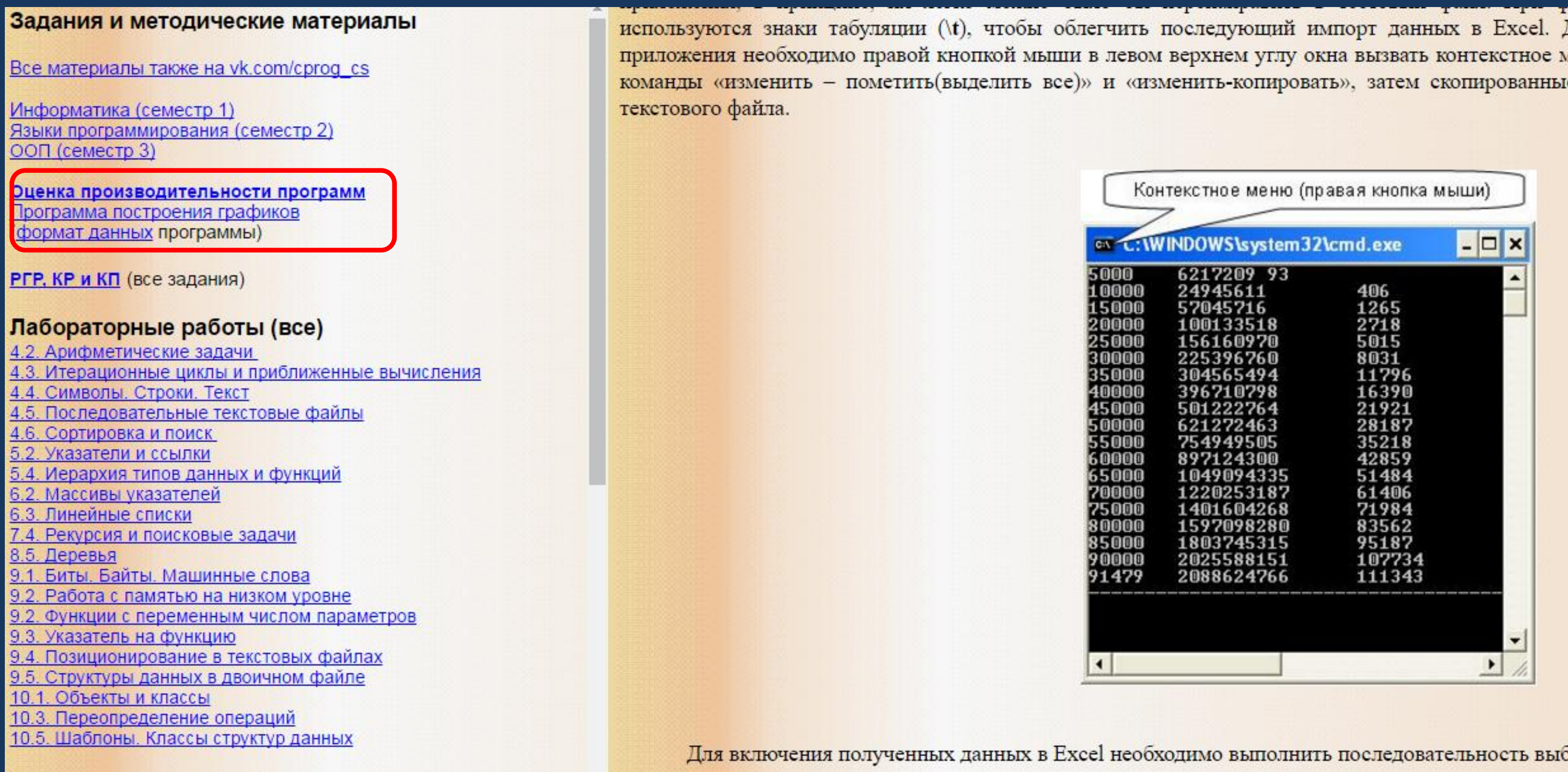
Начало: S=S(0)=x, n=1, Sn=x;

Вычисление степенного ряда

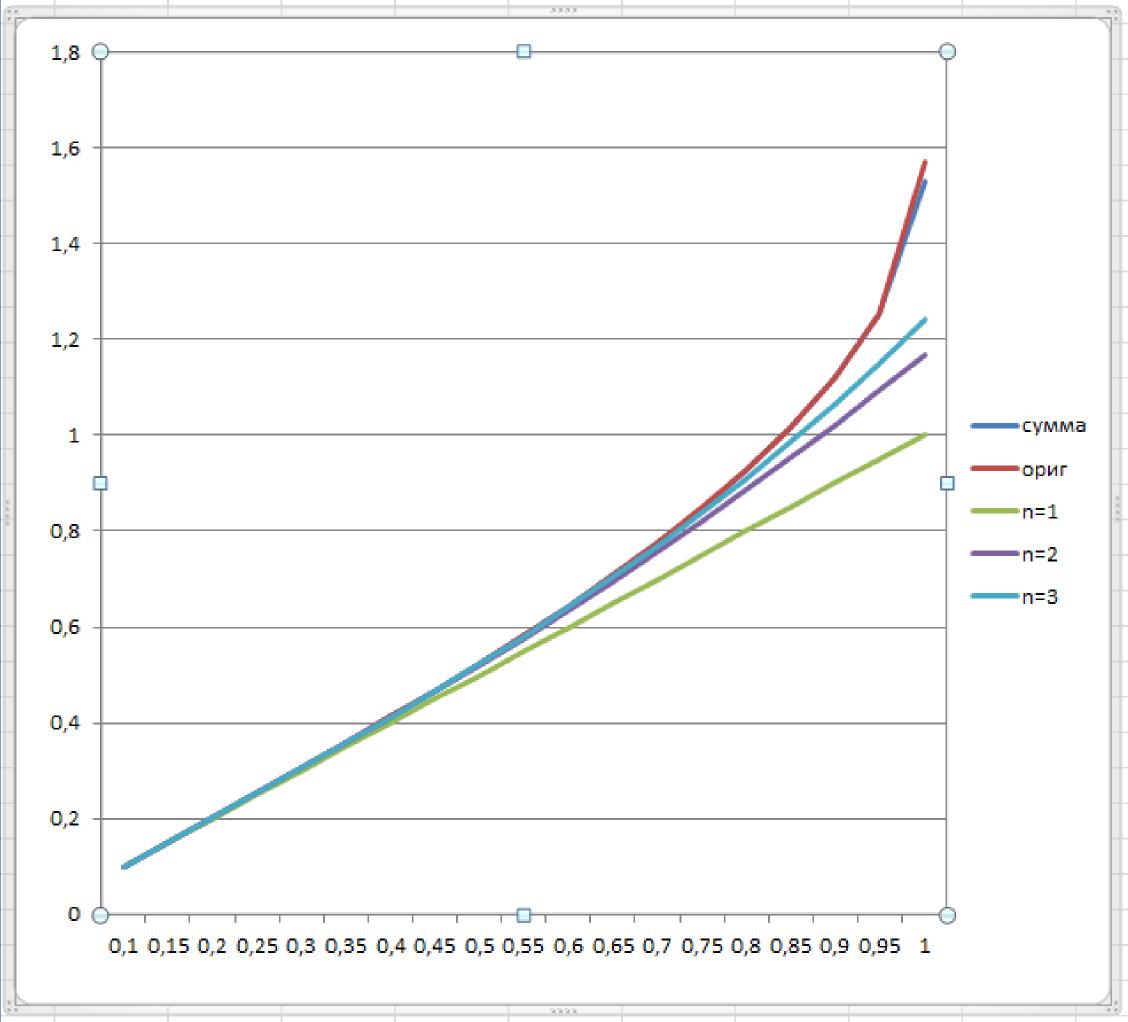
```
double sum(double x,double eps,int &n, int nMax){
double s,sn;                         // Сумма и текущее слагаемое ряда
     for (s=x, sn = x, n=1; n<nMax && fabs(sn) > eps; n++) {
     sn = sn * x * x * (2*n-1)*(2*n-1) / (2.*n * (2.*n + 1));
     s += sn;
     }
return s;}
// Вычисление степенного ряда для x в диапазоне от 0.1 до 1 с шагом 0.1
int main(){
double x,y;
int nn;
for (x=0.1; x <= 1.001; x += 0.05){
       y=sum(x,0.0001,nn,1000);
       printf("%d %0.2lf\t %0.4lf\t %0.4lf\n",nn,x,y,asin(x));
       }
return 0;
}
```

```
3  0.1     0.1002   0.1002
3  0.2     0.2014   0.2014
4  0.3     0.3047   0.3047
4  0.4     0.4115   0.4115
5  0.5     0.5236   0.5236
6  0.6     0.6435   0.6435
8  0.7     0.7754   0.7754
11  0.8    0.9272   0.9273
18  0.9    1.1195   1.1198
201  1.0   1.5310   1.5708
```

Вычисление степенного ряда



Вычисление степенного ряда



Точность приближения при различных n

- при x=1, оригинал asin(x) = nan
- при x>1 ряд расходится

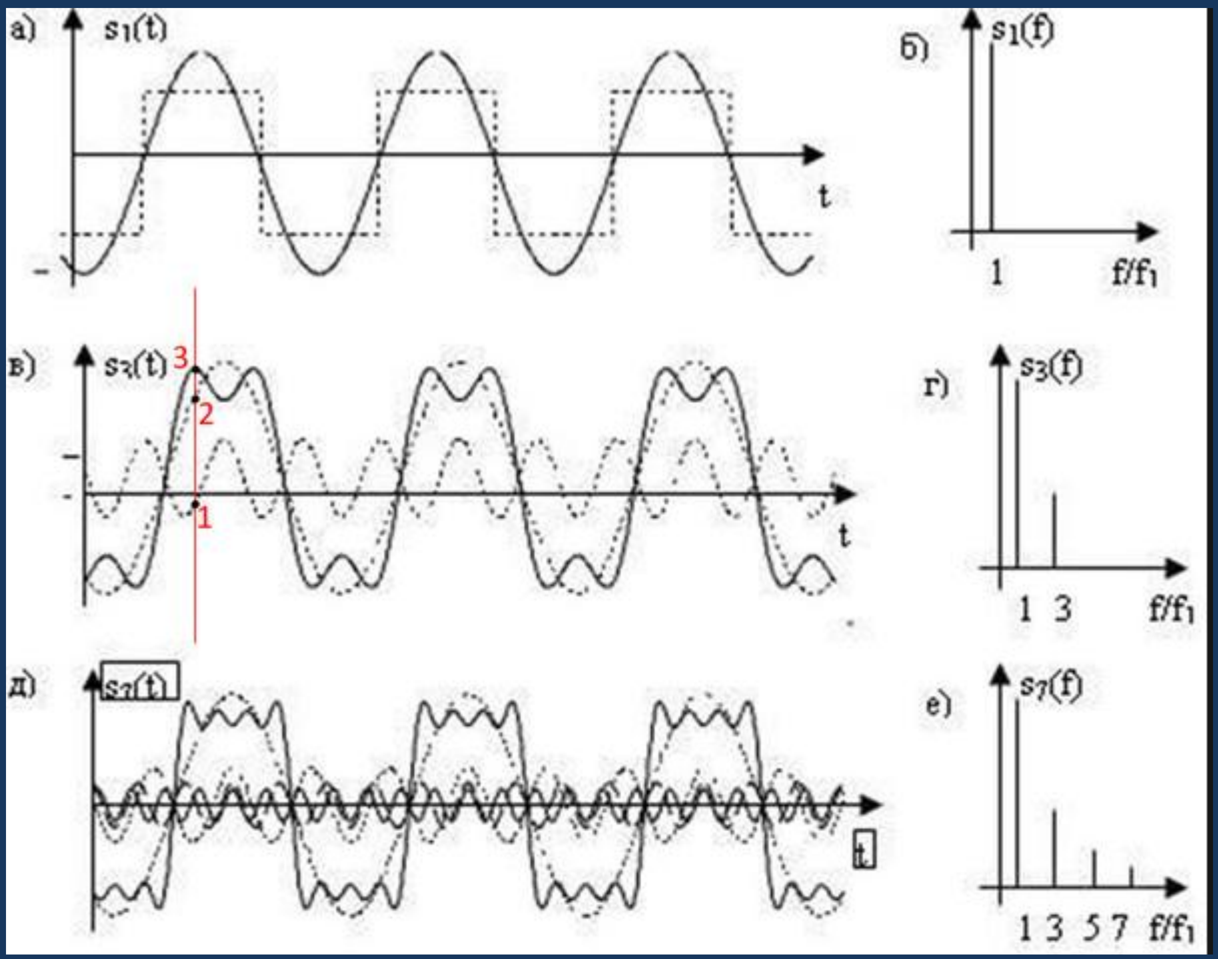
```
201 1.00         1.5310   nan
10000 1.05       inf      nan
```

```
201 1.00         1.5310   nan
1000 1.05        221982481455123152014457084531670253568.0000     nan
```

Вычисление степенного ряда

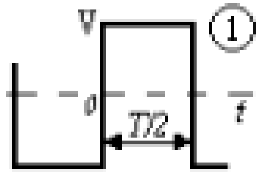
Вычисление гармонического ряда

Ряд Фурье: периодическая функция может быть разложена в виде суммы синусоид (спектр амплитудный и фазовый, дискретный). Непериодическая функция - непрерывный спектр



Вычисление гармонического ряда

Таблица 2

График $f(t)$	Ряд Фурье функции $f(t)$	Примечание
	$f(t)=\frac{4V}{\pi}\sum_{k=1}^{\infty}\frac{\sin k\omega t}{k}$	k=1,3,5,... $\omega=\frac{2\pi}{T}$
	$f(t)=\frac{8V}{\pi^2}\sum_{k=1}^{\infty}(-1)^{\frac{k-1}{2}}\frac{\sin k\omega t}{k^2}$	k=1,3,5,... $\omega=\frac{2\pi}{T}$
	$f(t)=\frac{4V}{\omega\tau\pi}\sum_{k=1}^{\infty}\frac{\sin k\omega\tau}{k^2}\sin k\omega t$	k=1,3,5,... $\omega=\frac{2\pi}{T}$
	$f(t)=\frac{V}{2}-\frac{V}{\pi}\sum_{k=1}^{\infty}\frac{1}{k}\sin k\omega t$	k=1,2,3,4,5 $\omega=\frac{2\pi}{T}$

Вычисление гармонического ряда

Рекуррентное вычисление кратных sin/cos

```
sin(nx) = sin((n-1)x)cos(x) + cos((n-1)x)sin(x)
cos(nx) = cos((n-1)x)cos(x) - sin((n-1)x)sin(x)
```

17	$\sin(x)+\sin(3x)/3^3+\sin(5x)/5^3+\ldots+\sin((2n-1)x)/(2n-1)^3$	$x(\pi - x)(\pi/8)$

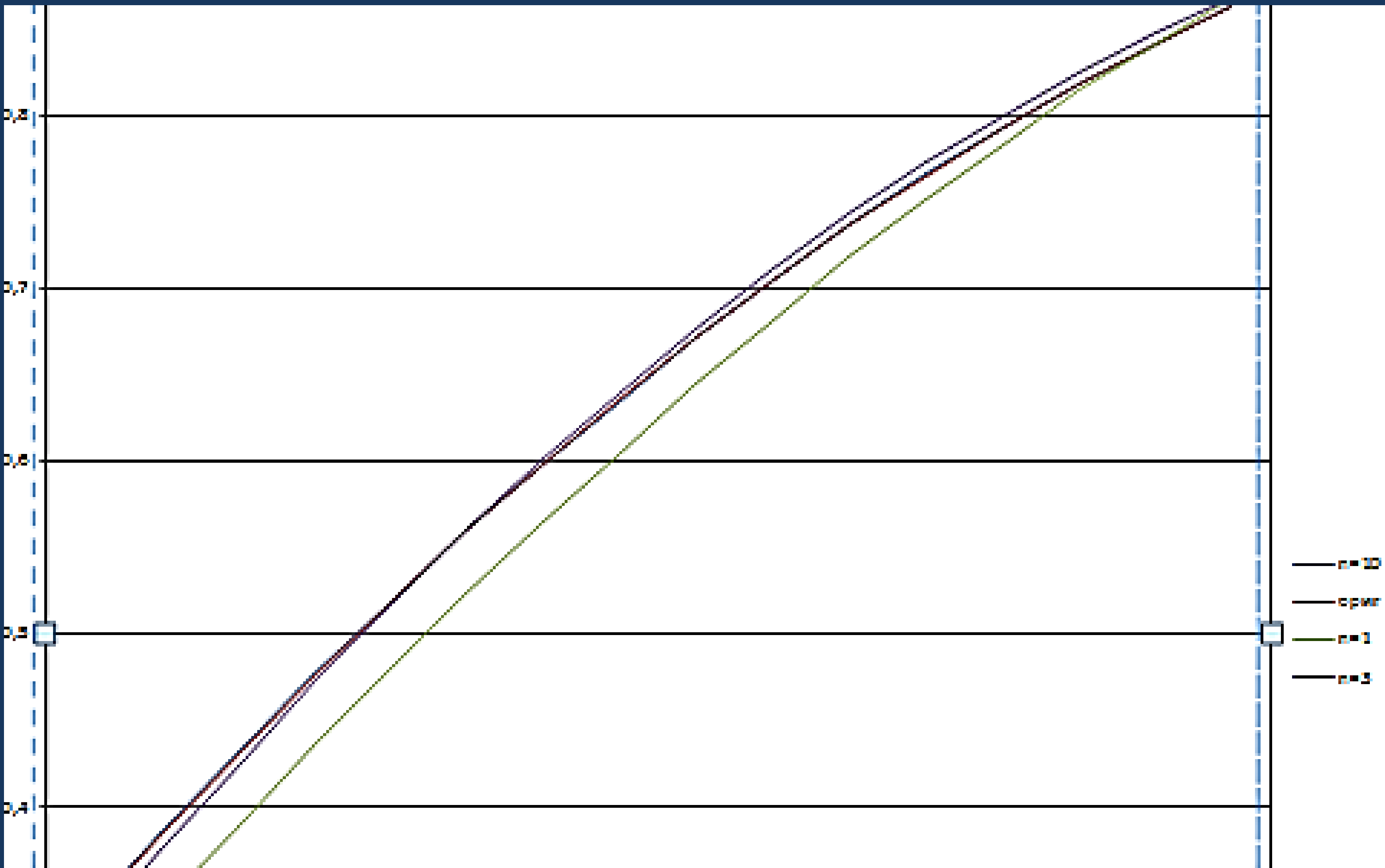
Количество шагов: решение уравнения 1/(2n-1)^3 = eps,
т.к. sin<=1

Вычисление гармонического ряда

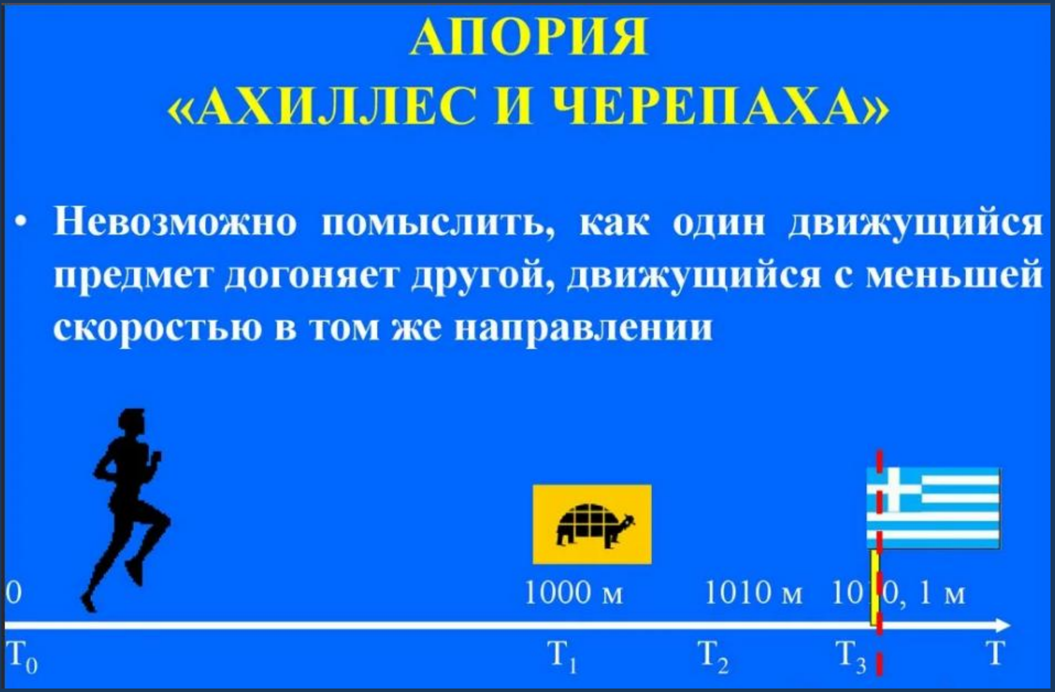
```
double sum2(double x,double eps,int &n, int nMax){
double Cn,Cn1,C0,Sn,Sn1,S0;     // S0=sin(x), Sn1=sin((n-1)x), Sn=sin(nx)
Cn1=C0=cos(x);                  // при n=2  sin((n-1)x)=sin(x)
Sn1=S0=sin(x);
int NN = pow(1/eps,1./3)/2;     // Оценка ДЛЯ sin()<=1, КРАТНЫЕ sin НЕМОНОТОННЫ
double S=0;
for (n=1;n<NN && n<nMax;n++){       // ВЫЧИСЛЯЮТСЯ ДЛЯ СЛЕДУЮЩЕГО N
   if (n%2==1)
       S += Sn1 / pow(n,3);              // n=1,3,5
   Sn=Sn1*C0 + Cn1*S0;                  // Рекуррентная формула
   Cn=Cn1*C0 - Sn1*S0;
   Cn1=Cn;                                          // Переход к следующему шагу
   Sn1=Sn;                                          // текущий становится предыдущим
   }
return S;
}

 int main(){
double x,y;
int nn;
double pi=3.1415926;
for (x=0.1; x <= pi+0.5; x += 0.05){
       y=sum2(x,0.0001,nn,1000);
       printf("%d %0.2lf\t %0.4lf\t %0.4lf\n",nn,x,y,x*(pi-x)*(pi/8));
       }
return 0;
}
```

Вычисление гармонического ряда

Вычисление гармонического ряда



PS: Ахиллес и черепаха, итерационный цикл, float/double, пределы и непрерывность



«Школьное» решение. Ахиллес все-таки догонит черепаху. **S** – начальное расстояние, **Sx** – расстояние, пройденное черепахой. **V1** – скорость Ахиллеса, **V2** – скорость черепахи. Равенство времен движения Ахиллеса и черепахи**: (S+Sx)/V1 = Sx/V2**, откуда
Sx = S*V2/(V1-V2)

Ахиллес и черепаха

Итеративное (предельное) решение. Описание апории в исходном варианте. Пока Ахиллес пробежит расстояние S, черепаха проползет
$S1 = S*V2/V1$
Пока Ахиллес пробежит расстояние S1, черепаха проползет
$S2 = S1*V2/V1 = S(V2/V1)^2$
Пока Ахиллес пробежит расстояние Sk, черепаха проползет
$Sk = Sk\text{-}1*V2/V1 = S(V2/V1)^k$
Тогда Sx –сумма бесконечного ряда **$S*\sum (V2/V1)^k$** при $k=1\ldots\infty$

Частный случай $V2/V1=0.1=1/10$ Оба решения выглядят так:
$Sx = S*V2/(V1-V2) = S*(1/9)$
$S*\sum (V2/V1)^k = S*(0.1 + 0.01 + 0.001 +\ldots.) =$
$S*0.1111111111111111111111111\ldots$

Первое решение – точное значение **1/9**,
Второе решение – то же самое значение в виде бесконечной десятичной дроби, для которого первый вариант является **пределом**

Аналогии: Первое решение – вещественное число в математике, второе – формат float/double при конечном числе разрядов