Тема 4. ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ИТЕРАЦИОННОЙ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Цель занятия: закрепление навыков в выборе и использовании операторов цикла; овладение практическими навыками разработки и программирования алгоритмов итерационных процессов.

Теоретические сведения

Циклы с неизвестным заранее числом повторений, которые выполняются, пока истины определенные условия, называются *итерационными*.

Условиями окончания цикла могут быть: достижение заданной точности вычисления результата, изменение знака определенной величины и т.д.

Для итерационного цикла известно условие выполнения цикла.

Задача 4.1. Дана последовательность целых чисел, за которой следует 0. Найти минимальный элемент этой последовательности.

```
void main()
{
	int a, min;
	cout<<"\nEnter a\n";
	cin>>a;
	min=a;
	while (a!=0) //for(;a!=0;)
	{
		cout<<"\nEnter a\n";
		cin>>a;
		if (a!=0&&a<min) min=a;
	}
	cout<<"\nmin="<<min;
}
```

Тесты приведены в табл. 8.1.

Таблица 4.1

Тесты к примеру 4.1

Последовательность *a*	min число
2 55 –3 –10 0	–10
12 55 4 27 0	4
[illegible]	[illegible]

Задача 4.2. Найти сумму чисел Фибоначчи, меньших заданного числа Q.

Примечание. Последовательность чисел Фибоначчи u_0, u_1,… образуется по закону $u_0=0$, $u_1=1$, $u_i=u_{i-1}+u_{i-2}$ $(i=2,3,..)$.

```
void main()
{
	int u0=0,u1=1, s=1,Q,ui;
	cout<<"\nEnter Q\n";
	cin>>Q;
	if(Q<=0)cout<<"Error in Q";
	else
		if(Q==1)cout<<"\nS=0";
		else
		{
			ui=u0+u1;
```

```
            while(ui<Q)             //for(;ui!=0;)
            {
                s+=ui;
                u0=u1;
                u1=ui;
                ui=u0+u1;
            }
            cout<<"\nS="<<s;
        }
    getch();
}
```

Тесты приведены в табл. 4.2.

Таблица 4. 2

Тесты к примеру 8.2

Значение числа Q	Сумма s
-1	Error in Q
0	Error in Q
1	0
2	2
3	4
10	20

Алгоритм, в состав которого входит итерационный цикл, называется ***итерационным алгоритмом***. Итерационные алгоритмы используются при реализации итерационных численных методов.

В итерационных алгоритмах необходимо обеспечить обязательное достижение условия выхода из цикла (сходимость итерационного процесса). В противном случае произойдет зацикливание алгоритма, т.е. не будет выполняться основное свойство алгоритма — результативность.

Вычисление суммы бесконечного ряда с заданной точностью является типичной задачей, использующей итерационный цикл, так, как заранее неизвестно, при каком члене ряда будет достигнута требуемая точность [2].

Пусть бесконечная сумма имеет вид: $s = \sum_{i=1}^{\infty} a_i$. Каждое слагаемое суммы является функцией от номера *i*, определяющего место этого слагаемого в сумме, а также может являться функцией одного или нескольких дополнительных параметров.

Вычисление суммы ряда состоит в получении в результате циклического процесса последовательности s_1, s_2, …, s_i, …, сходящейся к своему предельному значению. В общем случае начальное значение номера члена ряда *i* может быть отличным от 1 (например, равным 0). Суммирование считается законченным при выполнении условия достижения заданной точности ε: $|s_i - s_{i-1}| \le \varepsilon$ или $|a_i| \le \varepsilon$.

Каждый член суммы a_i вычисляется по формуле общего члена ряда.

Задача 4.3. Пусть x – некоторое число, а e = 0.001. Вычислить сумму элементов бесконечно убывающей знакопеременной последовательности $\{a_n\}$, где $a_n=(-1)^n(2x)^n/n!$, удовлетворяющих условию $|a_n|$ >e, n=1,2… . Определить количество слагаемых. Вывести на экран результаты вычислений.

```
#include <iostream>
#include <windows.h>
#include <math.h>
using namespace std;
```

```
int main()
{
  SetConsoleOutputCP(1251);
//Сначала сумма равна нулю, а факториал единице
  double  summa=0, x, a, e = 0.001;
  int    fact=1, n=1, z=-1;
  cout<<"Введите число x:\n";
  cin>>x;
  a =-2*x;
  while  (abs(a) > e)
          {summa = summa + a;
          n += 1;                              //Вычисляем an+1}
          z = -z;                             //Вычисляем (-1)n+1}
          fact *= n;                      //Вычисляем факториал}
          a = z * pow(2 * x, n) / fact;            //См. Замечание}
          }
cout<<"Сумма ="<< summa<<"\n";
cout<<"Количество слагаемых ="<< n-1;
return 0;
}
```

Замечание. Существует более экономный способ вычисления значения переменной *a* (элемента именно данной последовательности), используя предыдущее значение и команду присваивания вида *a*=*a***M*, где для данного примера *M*=-*x*/(*n*+1) (коэффициент рекуррентности). Реализуйте этот способ самостоятельно.

Задание 4.1. Решите задачу 4.1: 1) модифицируя программу, учитывая замечание; 2) модифицируя программу-пример при помощи следующей конструкции для вычисления знакопеременной суммы:

if (n % 2 !=0) summa=summa – a **else** summa = summa+a

Сравните результаты.

Если в формулу общего члена суммы входят степени и факториалы, то для уменьшения затрат времени на вычисление текущего члена ряда целесообразно использовать рекуррентную формулу.

Для получения рекуррентной формулы используется отношение текущего члена ряда к предыдущему: $\frac{a_i}{a_{i-1}}$.

Задача 4.4 Вычислить сумму ряда $z = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} - \ldots$ с точностью $\varepsilon = 10^{-4}$.

Для получения рекуррентной формулы (коэффициента рекуррентности) используем отношение текущего члена ряда к предыдущему:

$\frac{x^i}{i!} \Big/ \frac{x^{i-1}}{(i-1)!} = \frac{x^i \cdot (1 \cdot 2 \cdot \ldots \cdot (i-1))}{x^{i-1}(1 \cdot 2 \cdot \ldots \cdot (i-1) \cdot i)} = \frac{x}{i}$. Тогда $a_i = a_{i-1} \cdot \frac{x}{i}$. Точность будет достигнута, если $a_i < 10^{-4}$.

Схема алгоритма решения этой задачи приведена на рис. 4.1.

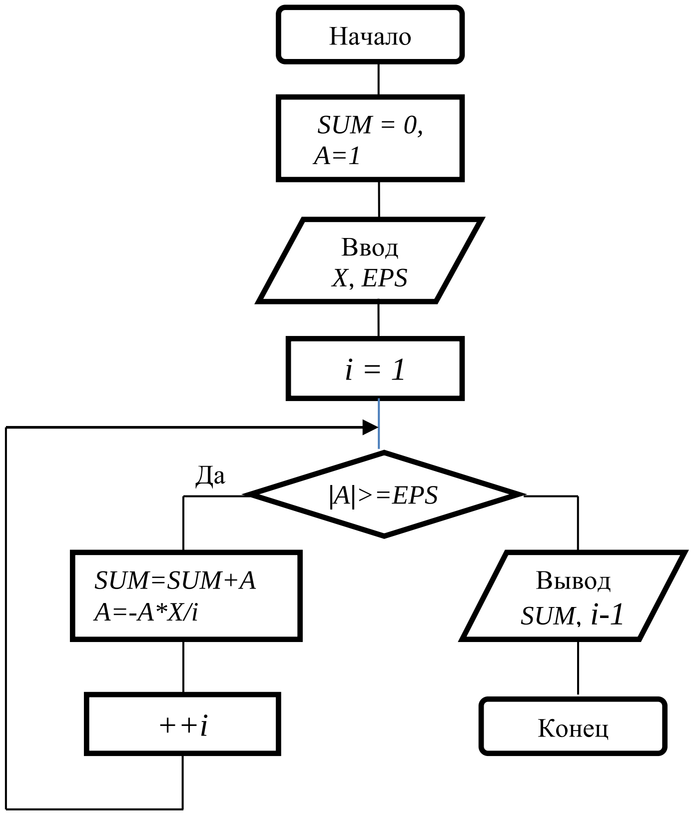


Рисунок 4.1 - Схема алгоритма решения задачи 4.4

Программа будет иметь вид:

```
#include <iostream>
#include <windows.h>
#include <math.h>
using namespace std;
int main()
{SetConsoleOutputCP(1251);
//Сначала сумма равна нулю
  double SUM = 0, X, EPS, A=1;
  int i;
  cout<<"Введите X\n";
  cin>>X;
  cout<<"Введите ТОЧНОСТЬ\n";
  cin>>EPS;
//Считать сумму членов ряда пока  |a_i| ≤ ε
  for ( i = 1; abs(A)>=EPS; ++i)
  {
      SUM=SUM+A;    // суммирование
      A=-A*X/i;  // следующий элемент ряда (X/i-коэффициент рекуррентности)
  }
  cout<<"СУММА="<< SUM;
  cout<<"\nКОЛИЧЕСТВО СЛАГАЕМЫХ ="<< i-1;
  getch();          // ждать нажатия любой клавиши
  return 0;
}
```

Пример 4.5. Вычислить значение суммы членов бесконечного ряда с заданной точностью ε. На печать вывести значение суммы и число членов ряда, вошедших в сумму. Для проверки полученного результата осуществить вызов функции, разложенной в бесконечный ряд.

Требуется вычислить сумму ряда $x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\dots$ (*sinx*) с точностью $\varepsilon=10^{-4}$.

Алгоритм 1. Каждый член суммы a_i вычисляется по формуле общего члена ряда. Схема алгоритма решения этой задачи приведена на рис. 4.2.

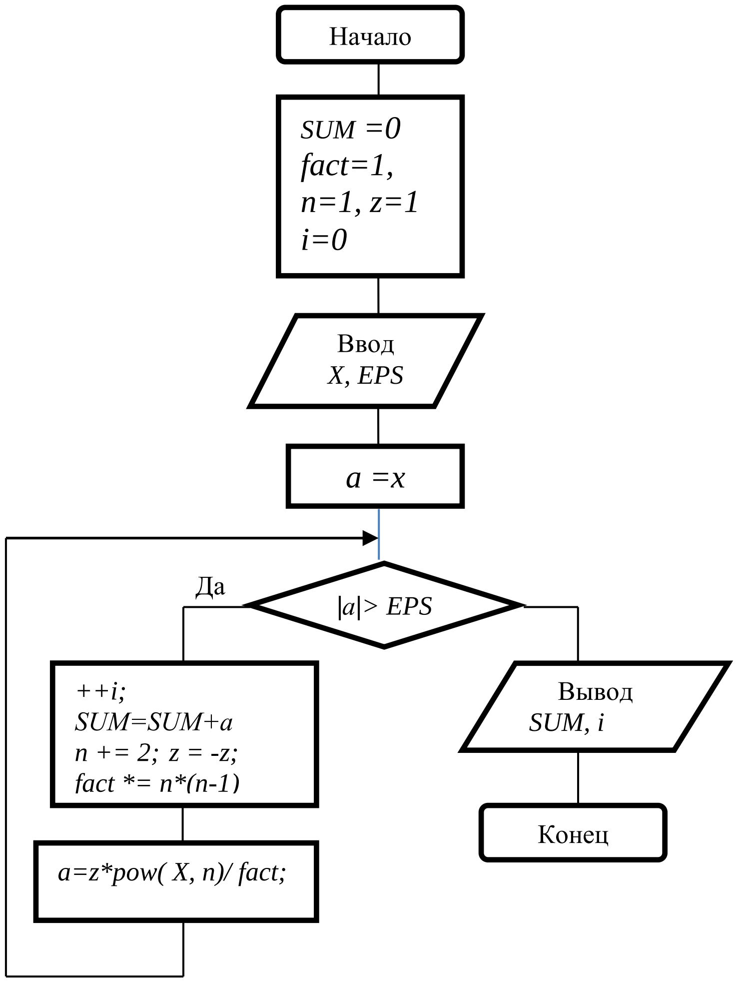


Рис. 4.2. Схема алгоритма решения примера 4.5 (Алгоритм 1)

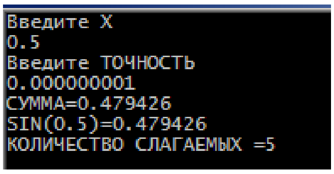
Программа будет иметь вид:

```
int main()
{
	SetConsoleOutputCP(1251);
//Сначала сумма равна нулю
	double SUM = 0, X, EPS, A;
	int   i=0,fact=1, n=1, z=1;
	cout<<"Введите X\n";
	cin>>X;
	A=X;
	cout<<"Введите ТОЧНОСТЬ\n";
```

```
	cin>>EPS;
//Считать сумму членов ряда пока
	while  (abs(A)>=EPS)
	{
		++i;			//число членов ряда, вошедших в сумму.
		SUM=SUM+A;	// суммирование
		n += 2;
		z = -z;					//Вычисляем новый знак
		fact *= n*(n-1);			//Вычисляем факториал
		A = z * pow( X, n) / fact;		//следующий элемент ряда

	}
	cout<<"СУММА="<< SUM;
	cout<<"\nSIN("<<X<<")="<<sin(X);
	cout<<"\nКОЛИЧЕСТВО СЛАГАЕМЫХ ="<< i;
	_getch();	// ждать нажатия любой клавиши
	return 0;
}
```

Результат:



Замечание. Вычисление знакопеременной суммы возможно при помощи следующей конструкции:

if (i % 2 !=0) summa=summa + a **else** summa = summa-a

Существует более экономный способ вычисления значения переменной *a* (элемента именно данной последовательности), используя предыдущее значение и команду присваивания вида *a*=*a***M* (*M* - коэффициент рекуррентности).

Алгоритм 2. Если в формулу общего члена суммы входят степени и факториалы, то для уменьшения затрат времени на вычисление текущего члена ряда целесообразно использовать рекуррентную формулу.

Для получения рекуррентной формулы используется отношение текущего члена ряда к предыдущему: $\frac{a_i}{a_{i-1}}$.

Для получения рекуррентной формулы (коэффициента рекуррентности) используем отношение текущего члена ряда к предыдущему:

$\frac{x^{2*i+1}}{(2i+1)!} \Big/ \frac{x^{2(i-1)+1}}{(2(i-1)+1)!} = \frac{x^2}{2*i(2*i+1)}$. Тогда $a_i = a_{i-1} \cdot \frac{x^2}{2*i(2*i+1)}$. Точность будет достигнута, если $a_i < 10^{-4}$.

Схема алгоритма решения этой задачи приведена на рис. 4.3.

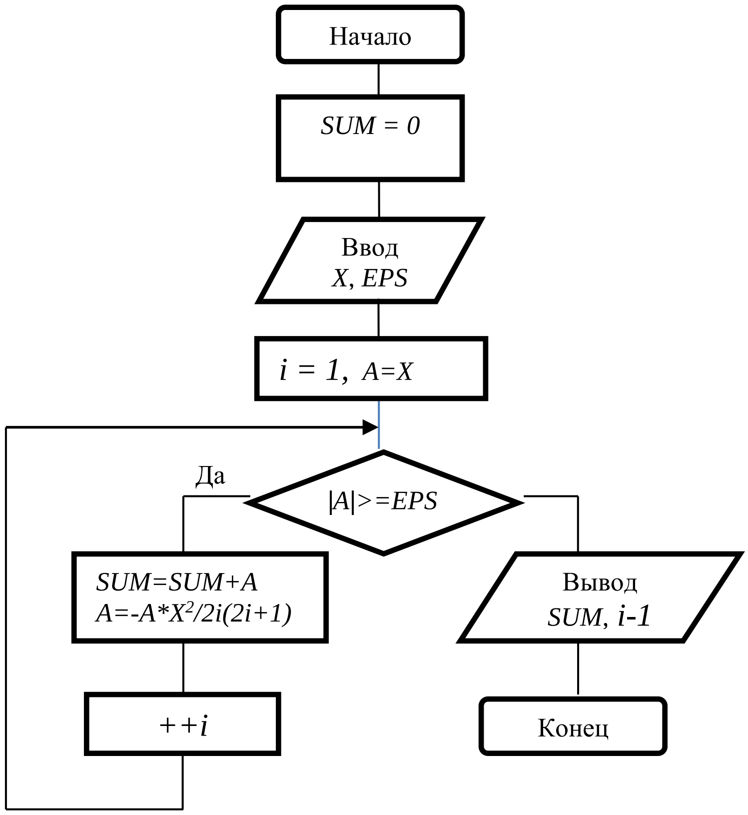


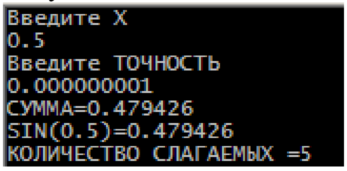
Рис. 4.3. Схема алгоритма решения примера 4.5 (Алгоритм 2)

Программа будет иметь вид:

```
int main()
{
	SetConsoleOutputCP(1251);
//Сначала сумма равна нулю
	double SUM = 0, X, EPS, A;
	int i;
	cout<<"Введите X\n";
	cin>>X;
	A=X;
	cout<<"Введите ТОЧНОСТЬ\n";
	cin>>EPS;
//Считать сумму членов ряда пока
	for ( i = 1; abs(A)>=EPS; ++i)
	{
		SUM=SUM+A;   // суммирование
		A=-A*X*X/(2*i*(2*i+1));       // следующий элемент ряда
	}
```

```
        cout<<"СУММА="<< SUM;
        cout<<"\nSIN("<<X<<")="<<sin(X);
        cout<<"\nКОЛИЧЕСТВО СЛАГАЕМЫХ ="<< i-1;
        _getch();      // ждать нажатия любой клавиши
        return 0;
}
```

Результат:



Нахождение корней уравнений

Аналитического решения многих алгебраических и трансцендентных уравнений получить не удается. Для решений таких уравнений используют приближенные итерационные методы (методы последовательных приближений). При этом решение уравнения разбивается на два этапа: 1) определение грубого значения корня, например, графическим путем; 2) уточнение значения корня.

Уточнение значения корня уравнения методами итерации и половинного деления требует организации цикла с неизвестным числом повторений. Предполагается, что известен интервал $a \le x \le b$, внутри которого существует один или несколько корней.

Сущность метода итераций заключается в том, что исходное уравнение представляется в виде $x = \varphi(x)$. Для практического применения метода итераций необходимо знать достаточные условия сходимости итерационного процесса: если в интервале между приближенным значением корня x_0 и значением корня уравнения x выполняется условие $|\varphi'(x)| < 1$, то метод дает возможность вычислить значение корня с заданной точностью ε. Если это условие не выполняется, то надо перейти к обратной функции. Новое значение корня вычисляется через предыдущее по формуле $x_i = \varphi(x_{i-1})$.

Повторяя этот процесс для x_1, x_2, …, можно вычислить корень с заданной точностью, которая определяется с помощью выражения $|x_i - x_{i-1}| \le \varepsilon$.

Задача 4.6. Методом итераций найти корень уравнения arcsin(2x+1)-x^2=0, расположенный на отрезке [-0,5; 0], с точностью $\varepsilon = 10^{-4}$. Вывести число итераций, необходимых для вычисления корня.

Заданное уравнение преобразуем к виду $x = \varphi(x)$ следующим образом:

arcsin(2x+1) = x^2;

sin (arcsin(2x+1)) = sin x^2;

2x+1 = sin x^2;

x = 0,5 (sin x^2-1).

Проверка условия сходимости метода итераций: $\varphi'(x) = x\cos x^2$. Очевидно, что $|\varphi'(x)| = |x\cos x^2| \le 0{,}5$ для всех $-0{,}5 \le x \le 0$. Следовательно, процесс итераций сходится.

Алгоритм решения уравнения по методу итераций представлен на рис. 4.4. Программа, реализующая схему алгоритма имеет вид:

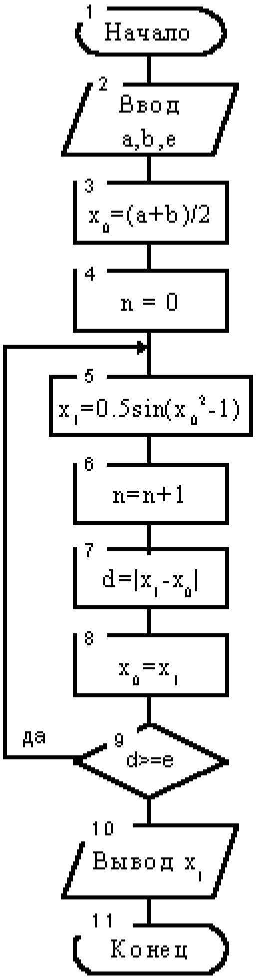


Рисунок 4.4 - Схема алгоритма решения задачи 4.6

```
int main()
{
  SetConsoleOutputCP(1251);
  double a,b,x1,x0,d,e;
  int n=0;
  cout<<"Введите a,b,e ";
  cin>>a>>b>>e;
  x0=(a+b)/2;
  do
            {  x1= 0.5*sin(x0*x0-1);
               n+=1;
               d=abs(x1-x0);
               x0=x1;}
   while (d>=e);
   cout<<"корень="<<x1<<"\n";
   cout<<"число итераций = "<<n;
  return 0;
}
```

Если функция имеет достаточно сложный вид, то проверка условия сходимости оказывается затруднительной. Для решения уравнения в этом случае можно использовать ***метод половинного деления***, который, хотя и требует значительного объема вычислений для медленно сходящихся функций, всегда приводит к искомому результату.

Сущность метода половинного деления заключается в следующем. Исходное уравнение представляется в виде $f(x) = 0$. Функция $f(x)$ непрерывна в интервале $[a,b]$, где отыскивается корень и $f(a)f(b)<0$, т.е. функция имеет разные знаки на концах интервала, причем $b–a> \varepsilon$. Для нахождения корня, принадлежащего отрезку $[a,b]$, отрезок делится пополам, т.е. выбирается начальное приближение $x = \frac{a+b}{2}$. Если $f(x) = 0$, то значение x является точным корнем уравнения $f(x) = 0$. Если $f(x) \neq 0$, то выбирается тот из отрезков $[a,x]$ или $[x,b]$, на концах которого функция $f(x)$ имеет противоположные знаки. Полученный отрезок снова делится пополам и повторяется то же рассмотрение.

Процесс деления отрезков продолжается до тех пор, пока длина отрезка, на концах которого функция имеет противоположные знаки, не станет меньше заданного числа ε.

Задача 4.7. Найти корень уравнения $e^x - x^2 = 0$ методом половинного деления на отрезке от -1до –0,5 с точностью до $\varepsilon = 10^{-4}$.

```
int main()
{
  SetConsoleOutputCP(1251);
  double a, b, x, e, fa, fb, fx;
   int nmax=4, n=0;
   cout<<"Введите a,b,e";
   cin>>a>>b>>e;
          fa = exp(a)-a*a;
          fb = exp(b)-b*b;
          if (fa*fb<0)
             {
              do
                 {    x=(a+b)/2;
                      fx = exp(x)-x*x;
                       if (fa*fx<0) b=x;
                          else  {  a=x; fa = fx; }
                       n++;
                  }
              while (abs(b-a)>=e || n<=nmax);
              cout<<"корень="<<x<<"\n";
             }
          else   cout<<"нет корня"<<"\n";
  return 0;
}
```

Методические указания

При подготовке к занятию необходимо: изучить организацию итерационных циклов; возможности языка С++ для организации таких циклов; освоить приемы программирования – уточнение корня уравнения, вычисление суммы членов бесконечного ряда, накопления суммы.

Аудиторные и домашние задания

1. Дано действительное A, найти из $1;1+\frac{1}{2};1+\frac{1}{2}+\frac{1}{3}...$, первое больше A.

2. Вычислить:y - первое из чисел sinx, sin sinx, sin sin sinx, … , меньшее по модулю 10^{-2}.

3. Дано вещественное число $b>0$, целое n. Последовательность a_1, a_2, ... образована по закону: $a_1=b$, $a_i=a_{i-1}-\frac{1}{\sqrt{i}}$, где $i=2,3,...,n$. Найти первый отрицательный член последовательности a_1, a_2, ...

4. Даны действительные числа a, b, ε ($a>b>0$, $\varepsilon>0$). Последовательности x_1, x_2, …, y_1, y_2, … образованы по закону: $x_1=a$, $y_1=b$, $x_k=\frac{1}{2}\left(x_{k-1}+y_{k-1}\right), y_k=\sqrt{x_{k-1}y_{k-1}}$. найти первое x_n такое, что $|x_n-y_n|<\varepsilon$.

5. Даны вещественные числа x, $\varepsilon\,(x\neq 0, \varepsilon>0)$. Вычислить с точностью ε бесконечную сумму и указать количество учтенных слагаемых (слагаемые, меньшие ε в сумму не включать) [1]:

а) $\sum_{i=1}^{\infty}\frac{x}{i^3+i\sqrt{|x|}+1}$;

б) $\sum_{k=1}^{\infty}\frac{1}{\sqrt{|x|+k^2}}$;

в) $\sum_{k=1}^{\infty}\frac{(-x)^{2k}}{2k!}$;

г) $\sum_{k=1}^{\infty}\frac{(-1)^k x^{k+2}}{(k+1)(k+2)!}$;

д) $z=\sin x-\frac{1}{2}\sin 2x+\frac{1}{3}\sin 3x-\ldots+\frac{1}{n}\sin nx-\ldots$;

е) $r=\frac{\cos 2x}{1\cdot 3}+\frac{\cos 4x}{3\cdot 5}+\ldots+\frac{\cos 2nx}{(2n-1)(2n+1)}+\ldots$

6. Не используя стандартные функции вычислить с точностью ε:

а) $y=shx=\frac{e^x-e^{-x}}{2}=x+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}+\ldots$;

б) $z=cosx=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots$;

в) $r=arctgx=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots\ (|x|<1)$;

г) $s=sin^2x=\frac{2\cdot x^2}{2!}-\frac{2^3\cdot x^4}{4!}+\frac{2^5\cdot x^6}{6!}-\ldots\ (x<1)$;

д) $p=e^{-x^2}=1-\frac{x^2}{1!}+\frac{x^4}{2!}-\frac{x^6}{3!}+\ldots(x<1)$.

7. Методом деления отрезка пополам вычислить корень уравнения вида $f(x)=0$, расположенный на интервале $[a,b]$, с погрешностью ε. Определить также число итераций, необходимое для нахождения корня:

а) $e^x-e^{-x}-2=0$, [0; 1];

б) $x^2-\ln(1+x)-3=0$, [2; 3];

в) $x^4+\cos x-2=0$, [0; 2];

г) $x^3+x^2+x+1=0$, [-2; 1];

д) $x-2+\sin\frac{1}{x}=0$, [1,2; 2].

8. Методом итераций вычислить корень уравнения вида $f(x)=0$, расположенный на интервале $[a,b]$, с погрешностью ε. Определить также число итераций, необходимое для нахождения корня:

а) $x^2+10x-10=0$, [0; 1];

б) $3x-4\ln x-5=0$, [2; 4];

в) $2x-3\ln x-3=0$, [0,5; 0,6];

г) $0{,}4+arctg\sqrt{x}-x=0$, [1; 2];

д) $x^3+x^2-3=0$, [1; 2].

Контрольные вопросы

1. Что такое итерационный цикл? Его отличия от цикла с заданным числом повторений.
2. Каково условие выхода из цикла при вычислении значения суммы бесконечного ряда?
3. Почему при вычислении значения текущего члена a_n используется простая переменная, а не индексированная?
4. Что такое рекуррентная формула? Каково ее назначение?
5. Какие два этапа необходимо выделить при нахождении корней уравнений?
6. В чем заключается сущность методов деления отрезка пополам и итераций при уточнении корня?
7. Каковы условия сходимости метода итераций?