Лабораторная работа 6

Численное интегрирование функции.

Цель работы. На примере разработки программы для численного интегрирования функции с заданной точностью методом прямоугольников и методом трапеций освоить следующие приемы программирования:

- передача в функцию параметров «по значению» и «по адресу»;
- передача в функцию имени функции;
- передача одномерных массивов в функцию;
- объединение разнородных данных в структуру;
- использование массивов из элементов типа структура;

Задание.

1. Численное интегрирование функции с заданной точностью методом прямоугольников.

Вычислить определённый интеграл $\int_a^b f(x)dx$ в пределах от ***a*** до ***b*** для четырех функций f1 = x, f2 = sin(22 * x), f3 = x^4 и f4 = arctg(x).

Вычисление интеграла оформить в виде функции IntRect.

Вычисления выполнить для пяти значений точности: 0.01, 0.001, 0.0001, 0.00001 и 0.000001.

Исследовать быстродействие алгоритма в зависимости от подынтегральной функции и требуемой точности (быстродействие алгоритма можно оценить числом элементарных прямоугольников ***n***).

Результаты представить в виде 5 таблиц, по одной таблице для каждого значения точности. В каждой таблице выводить данные для всех четырех функций.

Для печати таблицы результатов использовать функцию

void PrintTabl(I_print i_prn[],int k), приведенную в приложении 2.

Здесь i_prn[] – массив структур типа I_print размерностью k.

Вид таблицы приведен в Приложении 1.

2. Выполнить п.1, используя для интегрирования метод трапеций. Вычисление интеграла оформить в виде функции IntTrap.

Для печати таблиц результатов использовать ту же функцию, что и в методе прямоугольников.

Указания по выполнению работы.

Алгоритм метода Дарбу-Римана аналогичен алгоритму метода прямоугольников, только на каждом шаге вычисляются две суммы – верхняя (S2) и нижняя (S1):

```
f1 = f( x );          // значение функции на левой границе отрезка
f2 = f( x + dx );     // значение функции на правой границе
if(f1 <= f2 )          // возрастающий участок
{   S1 += f1 * dx;     // нижняя сумма
    S2 += f2 * dx;     // верхняя сумма
}
else                   // убывающий участок
{   S2 += f1 * dx;     // верхняя сумма
    S1 += f2 * dx;     // нижняя сумма
}
```

Вычисления прекращаются, если `|S2-S1| <eps.`

Задача вычисления определенного интеграла формулируется следующим образом: вычислить $\int_a^b f(x)dx$ для подынтегральной функции *f(x)* при заданных значениях пределов интегрирования ***a, b*** и требуемой точности ***eps***.

При численном интегрировании площадь под кривой заменяется суммой площадей «элементарных» прямоугольников с высотой, проведенной из середины основания.

Формула приближенного значения определенного интеграла представляется в виде

$$S = \sum_{i=1}^{N} f(x_i)\Delta x,$$

где: $x_i = a + \Delta x/2 + (i-1)\Delta x;$ *N* - число элементарных прямоугольников.

Для уменьшения объема вычислений множитель Δx следует вынести за знак суммы. Тогда в цикле нужно выполнять только суммирование, а затем полученную сумму один раз умножить на *Δx*.

Для оценки погрешности вычисления интеграла на практике используют правило Рунге. Суть правила состоит в том, что выполняют вычисление интеграла с двумя разными шагами изменения переменной *x*, а затем сравнивают результаты и получают оценку точности. Наиболее часто используемое правило связано с вычислением интеграла дважды: с шагом *Δx* и шагом *Δx/2*.

Для методов прямоугольников и трапеций погрешность $R_{\Delta x/2}$ вычисления интеграла с шагом *Δx/2* оценивается следующей формулой:

$$|R_{\Delta x/2}| = \frac{|I_{\Delta x/2} - I_{\Delta x}|}{3}, \qquad (1)$$

где $I_{\Delta x/2}$ – значение интеграла, вычисленное с шагом *Δx/2*; $I_{\Delta x}$ – значение интеграла, вычисленное с шагом *Δx*.

В программе вычисления интеграла с точностью ***eps*** во внутреннем цикле находят значение определенного интеграла с шагом *Δx/2*. Во внешнем цикле производится сравнение значений интегралов, вычисленных с шагами *Δx* и *Δx/2* соответственно. Если требуемая точность не достигнута, то число разбиений удваивается, а в качестве предыдущего значения интеграла берут текущее и вычисление интеграла выполняется при новом числе разбиений.

Вычисление интеграла оформить в виде функции IntRect, формальными параметрами которой являются:

f – имя интегрируемой функции,

a, b – границы интервала интегрирования,

eps – требуемая точность,

n – число прямоугольников, при котором достигнута требуемая точность (выходной).

Функция возвращает значение интеграла.

Прототип функции:

```
double IntRect(TPF f,double a,double b,double eps,int& n);
```

Здесь:

TPF – тип указателя на подынтегральную функцию:

```
typedef double (*TPF)(double);
```

Для хранения и печати результатов вычислений используйте структуру, элементами которой являются наименование функции, значения интеграла (точное и вычисленное в виде суммы) и число «элементарных» прямоугольников ***n***, при котором достигнута требуемая точность. Точные значения, полученные аналитически, нужны для оценки правильности результатов численного интегрирования.

Так как в лабораторной работе требуется выполнять вычисление интеграла для четырех функций, для пяти значений точности для каждой функции и двумя методами, то для сокращения объема программы следует использовать циклы, а для обеспечения возможности реализации циклов обрабатываемые данные нужно хранить в массивах (массив указателей на функции, массив значений точности, массив структур для хранения и печати результатов вычислений).

Алгоритм метода трапеций аналогичен алгоритму метода прямоугольников, только площадь элементарной трапеции вычисляется по формуле: $S_т$=dx*(f(x)+f(x+dx))/2.

При этом значения функций на границах внутренних отрезков при вычислении интеграла используются дважды, а на границах интервала [a,b] - только один раз.

Прототип функции:

```
double IntTrap(TPF f,double a,double b,double eps,int& n);
```

Формулы для вычисления точных значений интеграла:

$\int_a^b x dx$=(b*b - a*a)/2.0;

$\int_a^b sin(22x)dx$=(cos(a*22.0) - cos(b*22.0))/22.0;

$\int_a^b x^4 dx$=(b*b*b*b*b - a*a*a*a*a)/5.0;

$\int_a^b arctg(x)dx$=b*atan(b) - a*atan(a) - (log(b*b+1) - log(a*a+1))/2.0;

Примеры передачи в функцию в качестве параметров одномерных массивов и имен функций.

Массивы и функции передаются в функцию через указатели.

Имя массива является указателем на его нулевой элемент. Указатель «ничего не знает» о длине массива и длина массива должна передаваться в функцию как параметр.

Имя функции указывает на первую команду кода функции.

Передача одномерных массивов в функцию

```
#include <iostream.h>
int sum(int *a,int n);
int main() {
      int n;
      int a[]={1,2,3,4,5,6,7,8};
      n=sizeof(a)/sizeof(int);// Определение
      // размерности инициализированного массива
      cout<<"n="<<n<<endl;
      cout << sum(a,n) <<"\n";
      return 0;
}
int sum(int* a,int n) // В функцию передаются
                      // указатель на начало массива
                      //(имя массива a) и его размерность(n)
{
      int i,s=0;
      int k=sizeof(a);//k – размер указателя (4 байта)
      cout << "k=" << k << endl;
      for (i = 0; i < n; i++)
```

```
            s += a[i];
      return s;
}
```

Передача имен функций в качестве параметров

```
#include <iostream>
using namespace std;
/*для удобочитаемости программы определяется новый тип
(тип пользователя) PF - указатель на функцию, которая имеет
один параметр типа int и не возвращает никакого значения*/
typedef void (*PF)(int);
//Определение функции f1
void f1(PF pf)    //функция получает в качестве параметра
                  // указатель типа PF
{     pf(5);           //вызов функции через указатель    }
void f(int i)
{ cout << i<<endl; }
int main() {
      f1(f);  //Функция выведет на экран число 5
      return 0;
}
```

Приложение 1.

Пример вывода таблицы результатов

```
//////////////////////////////////////////////////////////////////////////////
Быстродействие алгоритма численного интегрирования в зависимости от функции
и требуемой точности (быстродействие характеризуется числом отрезков n[i])
//////////////////////////////////////////////////////////////////////////////
Область интегрирования функций:  -1<= x <= 3
      Точность вычислений = 0.1
```

Функция	Интеграл	IntSum	N[i]
y=x	4.0000000000	4.0000000000	90
y=sin(22x)	-0.0000142441	-0.0000202415	90
y=x^4	48.8000000000	48.8184356937	810
y=arctg(x)	2.1570201976	2.1517031831	90

Точность вычислений = 0.0100000000

Функция	Интеграл	IntSum	N[i]
y=x	4.0000000000	4.0000000000	270
y=sin(22x)	-0.0000142441	-0.0000141178	2430
y=x^4	48.8000000000	48.8002276075	7290
y=arctg(x)	2.1570201976	2.1517031831	90

Функция для печати таблицы результатов

```
struct I_print{   //данные для печати результатов интегрирования
      char* name;//название функции
      double i_sum;     //значение интегральной суммы
      double i_toch;    //точное значение интеграла
      int n;      //число разбиений области интегрирования
                  //при котором достигнута требуемая точность
};

void PrintTabl(I_print i_prn[],int k)
{
const int m=4;//число столбцов таблицы
int wn[m]={12,18,18,10};//ширина столбцов таблицы
char *title[m]={"Function","Integral","IntSum","N "};
int size[m];
for(int i=0;i<m;i++)
      size[i]=strlen(title[i]);
//шапка таблицы
cout<<char(218)<<setfill(char(196));
for(int j=0;j<m-1;j++)
      cout<<setw(wn[j])<<char(194);
cout<<setw(wn[m-1])<<char(191)<<endl;

cout<<char(179);
for(int j=0;j<m;j++)
      cout<<setw((wn[j]-size[j])/2)<<setfill(' ')<<' '<<title[j]
      <<setw((wn[j]-size[j])/2)<<char(179);
cout<<endl;
for(int i=0;i<k;i++)
      {//заполнение таблицы
cout<<char(195)<<fixed;
for(int j=0;j<m-1;j++)
      cout<<setfill(char(196))<<setw(wn[j])<<char(197);
cout<<setw(wn[m-1])<<char(180)<<setfill(' ')<<endl;

cout<<char(179)<<setw((wn[0]-strlen(i_prn[i].name))/2)<<' '<<i_prn[i].name
                              <<setw((wn[0]-
strlen(i_prn[i].name))/2)<<char(179);
cout<<setw(wn[1]-1)<<setprecision(10)<<i_prn[i].i_toch<<char(179)
      <<setw(wn[2]-1)<<i_prn[i].i_sum<<setprecision(6)<<char(179)
      <<setw(wn[3]-1)<<i_prn[i].n<<char(179)<<endl;
      }
//низ таблицы
cout<<char(192)<<setfill(char(196));
for(int j=0;j<m-1;j++)
      cout<<setw(wn[j])<<char(193);
cout<<setw(wn[m-1])<<char(217)<<setfill(' ')<<endl;
      }
```