МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Юго-Западный государственный университет

(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ

Проректор по учебной работе

О.Г. Локтионова

«__» __________ 2016 г.

ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Методические указания по выполнению лабораторной работы

для студентов направления подготовки 09.03.01

Курск 2016

УДК 621.3
Составитель: Э.И. Ватутин

Рецензент
Кандидат технических наук, доцент *В.С. Панищев*

Программирование линейных алгоритмов: методические указания по выполнению лабораторных работ по дисциплине «Программирование» / Юго-Зап. гос. ун-т; сост.: Э.И. Ватутин; Курск, 2016. 10 с.: ил. 2.

Методические рекомендации содержат сведения по разработке линейных программ на современных языках программирования высокого уровня.

Предназначены для студентов направления подготовки 09.03.01 «Информатика и вычислительная техника».

Текст печатается в авторской редакции

Подписано в печать _________. Формат 60x84 1/16.
Усл. печ. л.   Уч. – изд.л.   Тираж 30 экз. Заказ   . Бесплатно.
Юго-Западный государственный университет
305040, Курск, ул. 50 лет Октября, 94.

Содержание

Введение

Целью работы является получение практических навыков при программировании арифметических выражений с использованием операций и подпрограмм стандартной библиотеки и линейных программ с их использованием.

Линейные алгоритмы представляют собой простейший класс алгоритмов, в котором действия (операторы) выполняются последовательно друг за другом, ветвления вычислительного процесса при этом не предусмотрено. При программировании подобных алгоритмов как правило используется оператор присваивания в совокупности с набором леводопустимых выражений (обычно переменных, в которые записывается результат) и праводопустимых арифметических выражений, включающих в своем составе в общем случае арифметические операции и вызовы подпрограмм стандартной библиотеки используемой среды разработки. Порядок вычисления операций в составе арифметического выражения определяется приоритетами операций и, при необходимости, может быть изменен с использованием круглых скобок «(» и «)».

Примером линейного алгоритма является нахождение суммы заданного небольшого количества слагаемых ряда Тейлора:

$$f(x)=f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\ldots+\frac{f^{(n)}(a)}{n!}(x-a)^n+\ldots=$$
$$=\sum_{k=0}^{\infty}\frac{f^{(k)}(a)}{k!}(x-a)^k$$

При этом правая часть ряда Тейлора записывается в виде арифметического выражения, что обеспечивает эффективное вычисление заданного значения (при большом количестве слагаемых эффективной является программная реализация с использованием циклов). Известно, что

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

причем с ростом числа слагаемых n абсолютная

$$\varepsilon = \left| f - f^* \right|$$

и относительная погрешности

$$\delta = \left| \frac{f - f^*}{f} \right| \cdot 100\%$$

убывают. Здесь $f = f(x)$ – точное значение функции, $f^* = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k$ – приближенное значение функции.

Известно, что многие математические функции можно представить в виде суммы бесконечного степенного ряда. В некоторых случаях это дает ряд преимуществ: можно считать приближенные значения при отсутствии точных, проще проводить операции интегрирования и дифференцирования и т.д. В программировании достаточно часто возникает вопрос о вычислении приближенной суммы первых n слагаемых подобного ряда (вычисления суммы бесконечного количества слагаемых потребовало бы бесконечного количества времени, поэтому на практике ограничиваются неким количеством n, которое позволяет аппроксимировать требуемую величину с заданной точностью). Подобная сумма называется n-ой частичной суммой.

Линейные программы для вычисления суммы первых *n* слагаемых ряда Тейлора

Попытаемся установить, что при увеличении значения n точность вычислений возрастает, на примере следующего выражения

$$\ln 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \ldots$$

Ограничимся вычислением первых пяти частичных сумм. Программа, выполняющая требуемые действия на Delphi, может быть оформлена в следующем виде:

```
{ Ln 2 = 1 – 1/2 + 1/3 – 1/4 + ... }
program Lr2_Simple_example;
{$APPTYPE CONSOLE}

uses
  SysUtils;

var
  PrecValue: Double;    { Точное значение (левая часть равенства) }
  ApprValue: Double;    { Приближенное значение (i-ая частичная сумма) }
  AbsE, RelE: Double;   { Погрешности }

begin
  { Вычисление точного значения }
  PrecValue := Ln(2);
  Writeln('Ln 2 = ', PrecValue:15:10);

  { Вычисление первой частичной суммы }
  ApprValue := 1;
  AbsE := Abs(ApprValue - PrecValue);
  RelE := AbsE / PrecValue * 100;
  Writeln('S1 =   ', ApprValue:15:10, '  e = ', AbsE:5:10, '  d = ', RelE:5:10, '%');

  { Вычисление второй частичной суммы }
  ApprValue := 1 - 1/2;
  AbsE := Abs(ApprValue - PrecValue);
  RelE := AbsE / PrecValue * 100;
  Writeln('S2 =   ', ApprValue:15:10, '  e = ', AbsE:5:10, '  d = ', RelE:5:10, '%');

  { Вычисление третьей частичной суммы }
  ApprValue := 1 - 1/2 + 1/3;
  AbsE := Abs(ApprValue - PrecValue);
  RelE := AbsE / PrecValue * 100;
  Writeln('S3 =   ', ApprValue:15:10, '  e = ', AbsE:5:10, '  d = ', RelE:5:10, '%');

  { Вычисление четвертой частичной суммы }
  ApprValue := 1 - 1/2 + 1/3 – 1/4;
  AbsE := Abs(ApprValue - PrecValue);
  RelE := AbsE / PrecValue * 100;
  Writeln('S3 =   ', ApprValue:15:10, '  e = ', AbsE:5:10, '  d = ', RelE:5:10, '%');

  { Вычисление пятой частичной суммы }
  ApprValue := 1 - 1/2 + 1/3 – 1/4 + 1/5;
  AbsE := Abs(ApprValue - PrecValue);
  RelE := AbsE / PrecValue * 100;
  Writeln('S3 =   ', ApprValue:15:10, '  e = ', AbsE:5:10, '  d = ', RelE:5:10, '%');

  Readln;
end.
```

Результаты выполнения программы представлены на рис. 1.

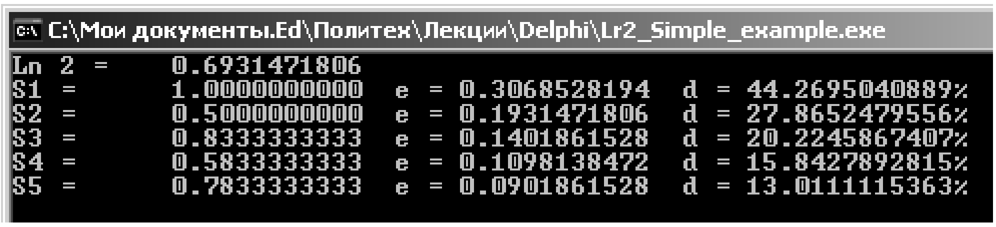

Рис. 1. Результаты выполнения программы

Видно, что ряд достаточно медленно сходится к точному значению, о чем можно судить по уменьшению погрешностей.

Рассмотрим еще один пример ряда Тейлора:

$$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

При вычислении приближенного значения ряда также ограничимся первыми пятью частичными суммами. Соответствующая программа на языке C++ приведена ниже.

```
#include <iostream>

using namespace std;

void main()
{
	double x;

	cout << "x = ";
	cin >> x;

	// Левая часть
	double L = sin(x);

	cout << "L = " << L << endl;

	// Правая часть (первая частичная сумма)
	double R = x;
	double e = fabs(L-R);
	double d = e / fabs(L) * 100;
	cout << "R = " << R << " e = " << e << " d = " << d << "%" << endl;

	// Вторая частичная сумма
	R = x - x*x*x/(1.0*2*3);
	e = fabs(L-R);
	d = e / fabs(L) * 100;
	cout << "R = " << R << " e = " << e << " d = " << d << "%" << endl;

	// Третья частичная сумма
	R = x - x*x*x/(1.0*2*3) + x*x*x*x*x/(1.0*2*3*4*5);
	e = fabs(L-R);
	d = e / fabs(L) * 100;
	cout << "R = " << R << " e = " << e << " d = " << d << "%" << endl;

	// Четвертая частичная сумма
	R = x - x*x*x/(1.0*2*3) + x*x*x*x*x/(1.0*2*3*4*5) - x*x*x*x*x*x*x/(1.0*2*3*4*5*6*7);
	e = fabs(L-R);
	d = e / fabs(L) * 100;
	cout << "R = " << R << " e = " << e << " d = " << d << "%" << endl;

	// Пятая частичная сумма
	R = x - x*x*x/(1.0*2*3) + x*x*x*x*x/(1.0*2*3*4*5) - x*x*x*x*x*x*x/(1.0*2*3*4*5*6*7) +
	    x*x*x*x*x*x*x*x*x/(1.0*2*3*4*5*6*7*8*9);
	e = fabs(L-R);
	d = e / fabs(L) * 100;
	cout << "R = " << R << " e = " << e << " d = " << d << "%" << endl;

	getchar();
}
```

Результаты выполнения программы приведены на рис. 2.

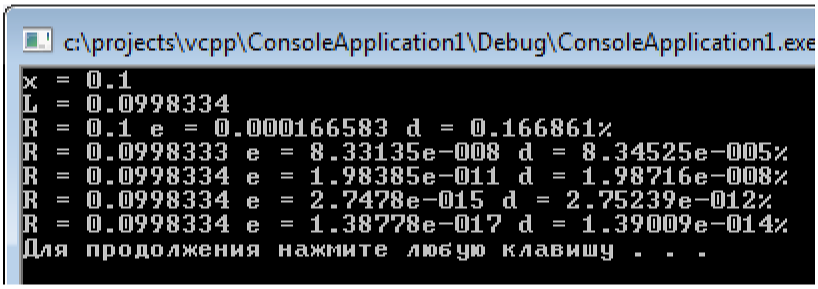

Рис. 2. Результаты выполнения программы

Для многих рядов Тейлора сходимость ряда обеспечивается в области значений аргумента $|x|<1$. Если данное условие не выполняется, правая часть ряда может не соответствовать левой.

Индивидуальные задания

В соответствии с индивидуальным вариантом произвести нахождение первых пяти частичных сумм ряда, вычислить абсолютную и относительную погрешности для каждой из них.

1. $\arcsin x = x + \frac{1}{2}\cdot\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\cdot\frac{x^5}{5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\cdot\frac{x^7}{7} + \ldots$

2. $\sqrt{1+x} = 1 + \frac{1}{2}x - \frac{1}{2\cdot 4}x^2 + \frac{1\cdot 3}{2\cdot 4\cdot 6}x^3 - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 8}x^4 + \ldots$

3. $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$

4. $\operatorname{sh} x = \frac{e^x - e^{-x}}{2} = \frac{x}{1!} + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots$

5. $\operatorname{ch} x = \frac{e^x + e^{-x}}{2} = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots$

6. $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$

7. $\ln\dfrac{1+x}{1-x}=2\left(x+\dfrac{x^3}{3}+\dfrac{x^5}{5}+\dfrac{x^7}{7}+...\right)$

8. $\ln(1+x)=x-\dfrac{x^2}{2}+\dfrac{x^3}{3}-\dfrac{x^4}{4}+...$

9. $\ln(1-x)=-\dfrac{x}{1}-\dfrac{x^2}{2}-\dfrac{x^3}{3}-\dfrac{x^4}{4}-...$

10. $\dfrac{1}{(1+x)^2}=1-2x+3x^2-4x^3+5x^4-...$

11. $\dfrac{1}{1+x}=1-x+x^2-x^3+...$

12. $\dfrac{1}{1-x}=1+x+x^2+x^3+...$

13. $1=\dfrac{1}{1\cdot 2}+\dfrac{1}{2\cdot 3}+\dfrac{1}{3\cdot 4}+...$

14. $\operatorname{arctg} x=x-\dfrac{x^3}{3}+\dfrac{x^5}{5}-\dfrac{x^7}{7}+...$

15. $\dfrac{1}{(x-1)^2}=1+2x+3x^2+4x^3+...$

16. $\dfrac{1-x^2}{(1+x^2)^2}=1-3x^2+5x^4-7x^6+...$

17. $\dfrac{2}{(1-x)^3}=1\cdot 2+2\cdot 3x+3\cdot 4x^2+4\cdot 5x^3+...$

18. $\dfrac{x}{(x-1)^2}=\dfrac{1}{x}+\dfrac{2}{x^2}+\dfrac{3}{x^3}+...$

19. $\dfrac{1}{2}\left(\operatorname{arctg} x-\dfrac{1}{2}\ln\dfrac{1-x}{1+x}\right)=x+\dfrac{x^5}{5}+\dfrac{x^9}{9}+\dfrac{x^{13}}{13}+...$

20. $\dfrac{\pi\sqrt{3}}{6}=1-\dfrac{1}{3\cdot 3}+\dfrac{1}{5\cdot 3^2}-\dfrac{1}{7\cdot 3^3}+...$

Содержание отчета

1. Титульный лист.
2. Индивидуальное задание.
3. Краткое описание стратегии решения.
4. Листинг программы.
5. Тестовые примеры, результаты тестирования.
6. Выводы.

Контрольные вопросы

1. Что представляют из себя линейные программы?
2. Для чего применяются ряды Тейлора?
3. Как производится расчет абсолютной и относительной погрешности при вычислении значения левой и правой части ряда Тейлора?
4. Что происходит с абсолютной и относительной погрешностями при увеличении числа слагаемых ряда?

Библиографический список

1. Емельянов С.Г., Ватутин Э.И., Панищев В.С., Титов В.С. Процедурно-модульное программирование на Delphi: учебное пособие. М.: Аргамак-Медиа, 2014. 352 с.
2. Зотов И.В., Ватутин Э.И., Борзов Д.Б. Процедурно-ориентированное программирование на C++: учебное пособие. Курск: КурскГТУ, 2008. 211 с.