МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

### федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Л.Н. Бариков

Отчёт

по лабораторной работе №11

по дисциплине ОСНОВЫ ПРОГРАММИРОВАНИЯ

на тему: «Передача имён функций в качестве параметров»

Работу выполнил

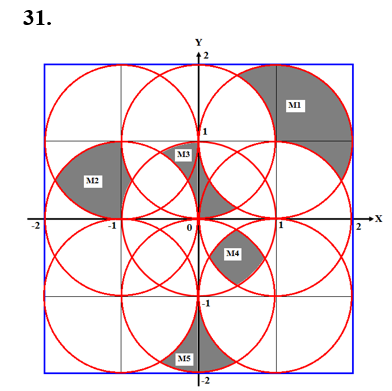
студент гр. 4141 В.С. Сыворотнев

Санкт-Петербург

2022

***Цель лабораторной работы:*** *изучение методов передачи имён функций в качестве параметров других функций; изучение методов нахождения приближённого значения определённого интеграла произвольной функции на заданном интервале и с заданной точностью; совершенствование навыков процедурного программирования на языке C/C++ при решении задач с использованием рекурсивных функций.*

***Задание на программирование:*** *используя технологию процедурного программирования реализовать заданный метод нахождения значения определённого интеграла произвольной функции на заданном интервале и с заданной точностью и применить его для определения площадей областей на плоскости в соответствии с индивидуальным заданием.*

***Вариант № 31***

***Математическая модель*** (условия принадлежности точек выделенным областям и аналитическое определение площадей выделенных областей)

- **условие 1** (принадлежность области *М*1):

x2 + (y – 1) 2> 1 {вне верхней средней окружности}

(x - 1) 2 + y 2> 1 {вне средней правой окружности}

(x *– 1)*2 +(*y* - 1)2<1 {внутри верхней правой окружности}

Площадь области М1:

Площадь четверти круга (*π\4) +* площадь М5 (0,3424) = π\4+0,3424=1,1278

Аналитический вид функций, подлежащих интегрированию:

интервал интегрирования [0.5, 2]; результат *s*1

; интервал интегрирования []; результат *s*2.

интервал интегрирования [0.5, 1]; результат *s*3

интервал интегрирования [1,], результат s4.

Площадь области М1=|*s*1|-|*s*2|-|*s*3|-|s4|.

- **условие 2** (принадлежность области *М*2):

x2 + (y - 1) 2> 1 {вне верхней средней окружности}

x2 + y2>1 {вне центральной окружности}

(x + 1) 2 + (y - 1) 2<1 {внутри левой верхней окружности}

(x + 1) 2 + y2 <1 {внутри средней левой окружности}

Площадь области М2:

площадь сектора 1200 (*πr*2/3) – площадь треугольника (√ 3/2×0.5) + площадь

квадрата (1×1) - площадь сектора 60 (πr2/6) - площадь треугольника (√3

/2×0.5)) = *π*/3–2\*(√ 3/4) + 1- *π*/6=1- (√ 3/2) + *π*/6= 0.6576

Аналитический вид функций, подлежащих интегрированию:

интервал интегрирования [, ]; результат s1

интервал интегрирования [, ]; результат s2

; интервал интегрирования [, -1]; результат *s*3.

интервал интегрирования [,-1]; результат *s*4.

Площадь области М2=|s1| - |s2| + |s3| - |s4|

- **условие 3** (принадлежность области *М*3):

*x*2 + *y*2 <1 {внутри центральной окружности}

(x + 1) 2 + y 2> 1 {вне средней левой окружности}

(x - 1) 2 + (y - 1) 2> 1 {вне верхней правой окружности}

*x*2+(*y* - 1)2< 1 {внутри средней верхней окружности}

Площадь области М3:

Площадь сектора 1800 (πr2 /2) – 2 \* (площадь сектора 1200 (πr2 /3) – площадь треугольника (√ 3 /2×0.5)) = π/2 – 2(π/3 - √3 /2×0.5) = √3 /2 - π/6=0,3424

Аналитический вид функций, подлежащих интегрированию:

интервал интегрирования [0.5, 0]; результат s1

интервал интегрирования [0, -0.5]; результат s2

; интервал интегрирования [0.5, 0]; результат *s*3.

интервал интегрирования [0, -0.5]; результат *s*4.

Площадь области М3=(|s1|+|s2|) - (|s3|+|s4|)

- **условие 4** (принадлежность области *М*4):

*x*2 + *y*2 <1 {внутри центральной окружности}

(*x –* 1)2 + *y*2 <1 {внутри правой средней окружности}

(*x –* 1)2 + (*y +* 1*)*2 <1 {внутри нижней правой окружности}

*x*2 + (*y +* 1*)*2 <1 {внутри нижней средней окружности}

Площадь области М4:

Площадь сектора 1200 (*πr*2/3) + площадь квадрата (1×1) – 4\*площади треугольника (√3/2×0.5) = *π*/3 + 1 - √3 = 0,3151

Аналитический вид функций, подлежащих интегрированию:

интервал интегрирования [0.5, ]; результат s1

интервал интегрирования [0.134, ]; результат s2

; интервал интегрирования [0.134, 0.5]; результат *s*3.

интервал интегрирования [0.5, ]; результат *s*4.

Площадь области М4=|s1|+|s2| - |s3|-|s4|

- **условие 5** (принадлежность области *М*5):

(x - 1) 2 + (y + 1) 2> 1 {вне нижней правой окружности}

(x + 1) 2 + (y + 1) 2> 1 {вне нижней левой окружности}

*x*2+(*y*+1)2<1 {внутри средней нижней окружности}

*y* < -1 {ниже линии *y*=-1}

Площадь области М5:

Площадь сектора 1800 (πr2 /2) – 2 \* (площадь сектора 1200 (πr2 /3) – площадь треугольника (√3 /2×0.5)) = π/2 – 2(π/3 - √3 /2×0.5) = √3 /2 - π/6=0,3424

Аналитический вид функций, подлежащих интегрированию:

интервал интегрирования [0.5, -0.5]; результат *s*1.

интервал интегрирования [0, -0.5]; результат *s*2.

интервал интегрирования [0.5, 0]; результат *s*3.

Площадь области М5=**|**s1**|**-(**|**s2**|** + **|**s3|)

***Текст программы***

#include<iostream>

#define \_USE\_MATH\_DEFINES

#include<cmath>

#define M\_Pi

#include<locale.h>

using namespace std;

typedef double(\* Tfun)(double);

double integral(double, double, int, double, Tfun, double);

double fun0(double x);

double fun1(double x);

double fun2(double x);

double fun3(double x);

double fun4(double x);

double fun5(double x);

double fun6(double x);

double fun7(double x);

double fun8(double x);

double fun9(double x);

double fun10(double x);

double fun11(double x);

double fun12(double x);

double fun13(double x);

double fun14(double x);

int main() {

  int i = 0, //номер области на рисунке

  n = 10;

  float x, y,            //координаты точки

  s1, s2, s3, s4, s5;        //площади областей

  double eps = .0001,                //точность вычисления

  s = 0;                    //вычисленная площадь области

  setlocale(LC\_ALL, "Russian");

  cout << "**\n**Введите координаты точки: x, y ";

  cin >> x >> y;

  if ((x - 1) \* (x - 1) + y \* y > 1 && (x - 1) \* (x - 1) + y \* y > 1 &&

      (x - 1) \* (x - 1) + (y - 1) \* (y - 1) < 1) {

    cout << "Точка в области M1. " << endl;

    i = 1;

  } else if ((y - 1) \* (y - 1) + x \* x > 1 && x \* x + y \* y > 1 &&

      (x + 1) \* (x + 1) + (y - 1) \* (y - 1) < 1 &&

      (x + 1) \* (x + 1) + y \* y < 1) {

    cout << "Точка в области М2. " << endl;

    i = 2;

  } else if ((x \* x) + y \* y < 1 && (x + 1) \* (x + 1) + y \* y > 1 &&

      (x - 1) \* (x - 1) + (y - 1) \* (y - 1) > 1 &&

      (x \* x) + (y - 1) \* (y - 1) < 1) {

    cout << "Точка в области М3. " << endl;

    i = 3;

  } else if ((x \* x) + y \* y < 1 && (x - 1) \* (x - 1) + y \* y < 1 &&

      (x - 1) \* (x - 1) + (y + 1) \* (y + 1) < 1 &&

      (x \* x) + (y + 1) \* (y + 1) < 1) {

    cout << "Точка в области М4. " << endl;

    i = 4;

  } else if ((y < -1) && (x - 1) \* (x - 1) + (y + 1) \* (y + 1) > 1 &&

      (x - 1) \* (x - 1) + (y + 1) \* (y + 1) > 1 &&

      (x \* x) + (y + 1) \* (y + 1) < 1) {

    cout << "Точка в области М5. " << endl;

    i = 5;

  } else cout << "Точка вне выделенных областей";

  cout.precision(4);        //число знаков после дес. точки

switch (i) {

    case 1: cout << "Formula: S1 = " << (s1 =M\_PI / 4. + (sqrt(3) / 2) - M\_PI / 6) << endl;

      cout << "Integral:S1 = " <<

           fabs((fabs(integral(.5, 2, n, eps, fun9, s))

           - fabs(integral(1 + sqrt(3) / 2, 2, n, eps, fun11, s))

           - fabs(integral(.5, 1, n, eps, fun12, s))

           - fabs(integral(1, 1 + sqrt(3) / 2, n, eps, fun10, s)))) << endl;

      break;

    case 2: cout << "Formula: S2 = " << (s2 = 1 - sqrt(3)/2 + M\_PI / 6) << endl;

      cout << "Integral:S2 = " <<

          fabs(integral((- sqrt(3) / 2), -1, n, eps, fun7, s) -

          fabs(integral((-sqrt(3) / 2), -1, n, eps, fun8, s) +

          fabs(integral((-1 - sqrt(3) / 2), -1, n, eps, fun14, s) -

          fabs(integral((-1 - sqrt(3) / 2), -1, n, eps, fun13, s))))) << endl;

      break;

    case 3: cout << "Formula: S3= " << (s3 = (sqrt(3) / 2) - M\_PI / 6) << endl;

      cout << "Integral:S3 = " <<

          fabs((integral(.5, 0, n, eps, fun7, s))

          + fabs(integral(0, -.5, n, eps, fun8, s)))

          - (fabs((integral(.5, 0, n, eps, fun11, s))

          + fabs(integral(0, -.5, n, eps, fun13, s)))) << endl;

      break;

    case 4: cout << "Formula: S4= " << (s4 =  M\_PI/3 + 1 - sqrt(3)) << endl;

      cout << "Integral:S4 = " <<

          fabs(integral(.5, (sqrt(3) / 2), n, eps, fun3, s)) +

          fabs(integral(.5, .134, n, eps, fun4, s)) -

          fabs(integral(.5, .134, n, eps, fun5, s)) -

          fabs(integral(.5, (sqrt(3) / 2), n, eps, fun6, s)) << endl;

      break;

    case 5: cout << "Formula: S5 = " << (s5 = (sqrt(3) / 2) - M\_PI / 6) << endl;

      cout << "Integral:S5 = " <<

          fabs(integral(.5, -.5, n, eps, fun0, s))

          - ((fabs(integral(0, -.5, n, eps, fun1, s))

          + fabs(integral(.5, 0, n, eps, fun2, s)))) << endl;

  }

  return 0;

}

//нахождение определенного интеграла методом криволинейных трапеций

double integral(double a, double b, int n, double eps, Tfun fun, double s1) {

  double x,

      h,

      s = 0;

  int i;

  h = (b - a) / n;

  for (i = 0; i < n; i++) {

    x = a + i \* h + h / 2;

    s += fun(x);

  }

  s \*= h;

  if (fabs(s - s1) > eps) {

    s = integral(a, b, 2 \* n, eps, fun, s);

  }

    return s;

}

double fun0(double x) {

  return (-sqrt(1 - x \* x) - 1);

}

double fun1(double x) {

  return (-sqrt(-2 \* x - x \* x) - 1);

}

double fun2(double x) {

  return (-sqrt(2 \* x - x \* x) - 1);

}

double fun3(double x) {

  return (-sqrt(1 - x \* x));

}

double fun4(double x) {

  return (-sqrt(2 \* x - x \* x));

}

double fun5(double x) {

  return (sqrt(2 \* x - x \* x) - 1);

}

double fun6(double x) {

  return (sqrt(1 - x \* x) - 1);

}

double fun7(double x) {

  return (-sqrt(1 - x \* x) + 1);

}

double fun8(double x) {

  return (sqrt(1 - x \* x));

}

double fun9(double x) {

  return (sqrt(2 \* x - x \* x) + 1);

}

double fun10(double x) {

  return (sqrt(2 \* x - x \* x));

}

double fun11(double x) {

  return (-sqrt(2 \* x - x \* x) + 1);

}

double fun12(double x) {

  return (sqrt(1 - x \* x) + 1);

}

double fun13(double x) {

  return (sqrt(-2 \* x - x \* x));

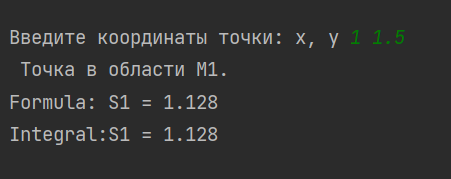
}

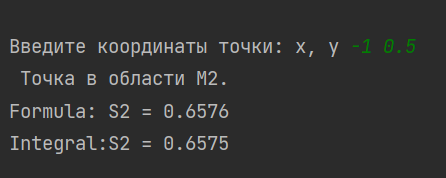
double fun14(double x) {

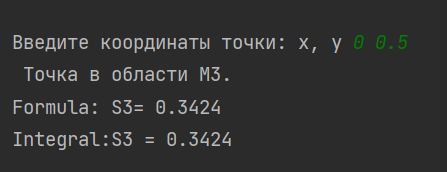
  return (-sqrt(-2 \* x - x \* x) + 1);

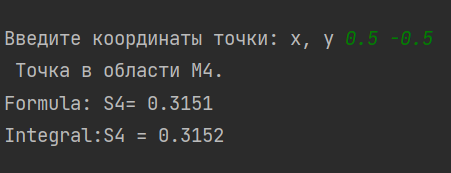
}

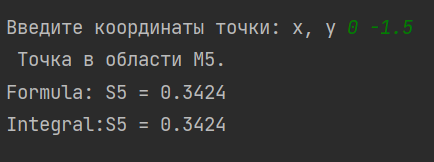
***Скриншот результатов выполнения программы***

****









***Вывод:*** *я**изучил методы передачи имён функций в качестве параметров других функций; изучил методы нахождения приближённого значения определённого интеграла произвольной функции на заданном интервале и с заданной точностью; усовершенствовал навыки процедурного программирования на языке C/C++ при решении задач с использованием рекурсивных функций.*