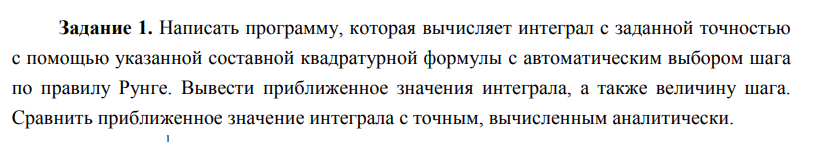
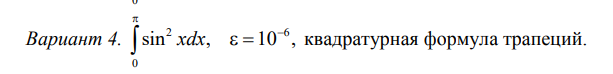
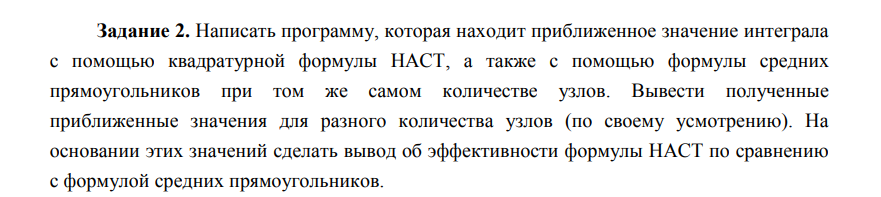
**Лабораторная работа № 2 «Численное интегрирование»**

Выполнил студент 3 курса 4 группы ФПМИ БГУ Видевич Александр.

**Задача.**

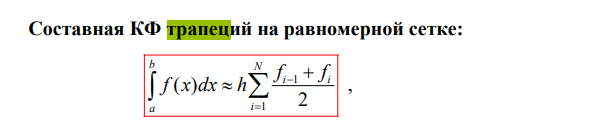




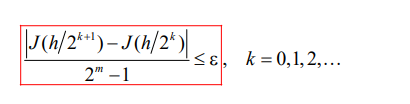


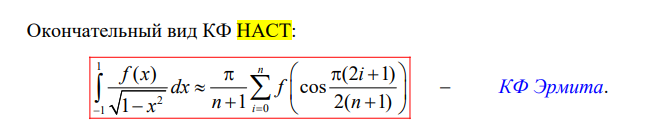


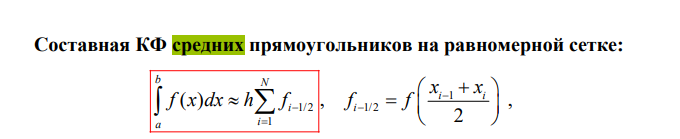
**Использованная теория.**



Правило Рунге







**Листинг программы**

*#include* <iostream>

*#include* <cmath>

double F1(double x)

{

*return* sin(x) \* sin(x);

}

double CalculateUsingTrapezoidalFormula(double a, double b, int N)

{

  double h = (b - a) / N;

  double integral = *0.5* \* (F1(a) + F1(b));

*for* (int i = *1*; i < N; ++i)

  {

    integral += F1(a + i \* h);

  }

*return* h \* integral;

}

double CalculateErrorUsingRungeRule(double I1, double I2, int p)

{

*return* std::abs(I1 - I2) / (pow(*2*, p) - *1*);

}

void RunTask1()

{

*const* double exactIntegral = M\_PI / *2.0*;

*const* double a = *0.0*;

*const* double b = M\_PI;

*const* double eps = *1e*-*6*;

*const* int p = *2*;

  int N = *1*;

  double error;

*do*

  {

*const* double I1 = CalculateUsingTrapezoidalFormula(a, b, N);

    N \*= *2*;

*const* double I2 = CalculateUsingTrapezoidalFormula(a, b, N);

    error = CalculateErrorUsingRungeRule(I1, I2, p);

  } *while* (error > eps);

*const* double stepSize = (b - a) / N;

  std::cout << "Приближенное значение интеграла: " << CalculateUsingTrapezoidalFormula(a, b, N) << '*\n*';

  std::cout << "Точное значение интеграла: " << exactIntegral << '*\n*';

  std::cout << "Шаг: " << stepSize << '*\n*';

}

double F2(double x)

{

*return* cos(x + *1*) / sqrt(*1* - x \* x);

}

double F2Numerator(double x)

{

*return* cos(x + *1*);

}

*// НАСТ*

double CalculateUsingHADA(double a, double b, int n)

{

  double sum = *0.0*;

*for* (int i = *0*; i <= n; i++)

  {

    sum += F2Numerator(cos(M\_PI \* (*2* \* i + *1*) / *2* / (n + *1*)));

  }

*return* sum \* M\_PI / (n + *1*);

}

double CalculateUsingReactangleFormula(double a, double b, int N)

{

*const* double h = (b - a) / N;

  double sum = *0.0*;

*for* (int i = *0*; i < N; i++)

  {

    sum += F2(a + i \* h + h / *2.0*);

  }

*return* h \* sum;

}

void RunTask2()

{

*const* double a = -*1.0*;

*const* double b = *1.0*;

  std::cout << "Количество узлов*\t*НАСТ*\t*формула средних прямоугольников*\n*";

*for* (int n = *1*; n <= *20*; n++)

  {

    double nystrom\_result = CalculateUsingHADA(a, b, n);

    double rectangle\_result = CalculateUsingReactangleFormula(a, b, n);

    std::cout << n << "*\t\t\t*" << nystrom\_result << "*\t*" << rectangle\_result << "*\n*";

  }

}

int main()

{

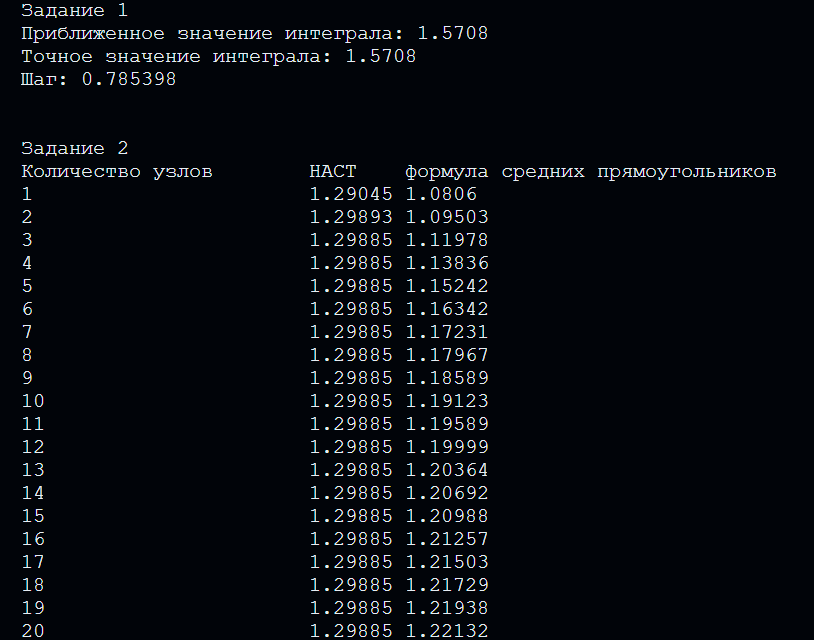
  RunTask1();

  RunTask2();

*return* *0*;

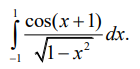
}

**Выходные данные.**



**Выводы**

В задании 1 точное значение интеграла совпадает с приближенным, причем программе хватило N = 1

В задании 2 приближенные значения, полученные по формуле НАСТ больше приближенны к точному значению, чем значения, полученные по формуле средних прямоугольников (для получения значения в 1.29885 потребовалось около 10000000 узлов). Это свидетельсвует об большей эффективности формулы НАСТ для вычисления интеграла .