МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №3**по дисциплине: «Вычислительная математика»

Выполнил: ст. группы ПВ-211

Медведев Дмитрий Сергеевич

Проверила:

Бондаренко Т.В.

Белгород 2023 г.

Численное интегрирование

# Вариант 8

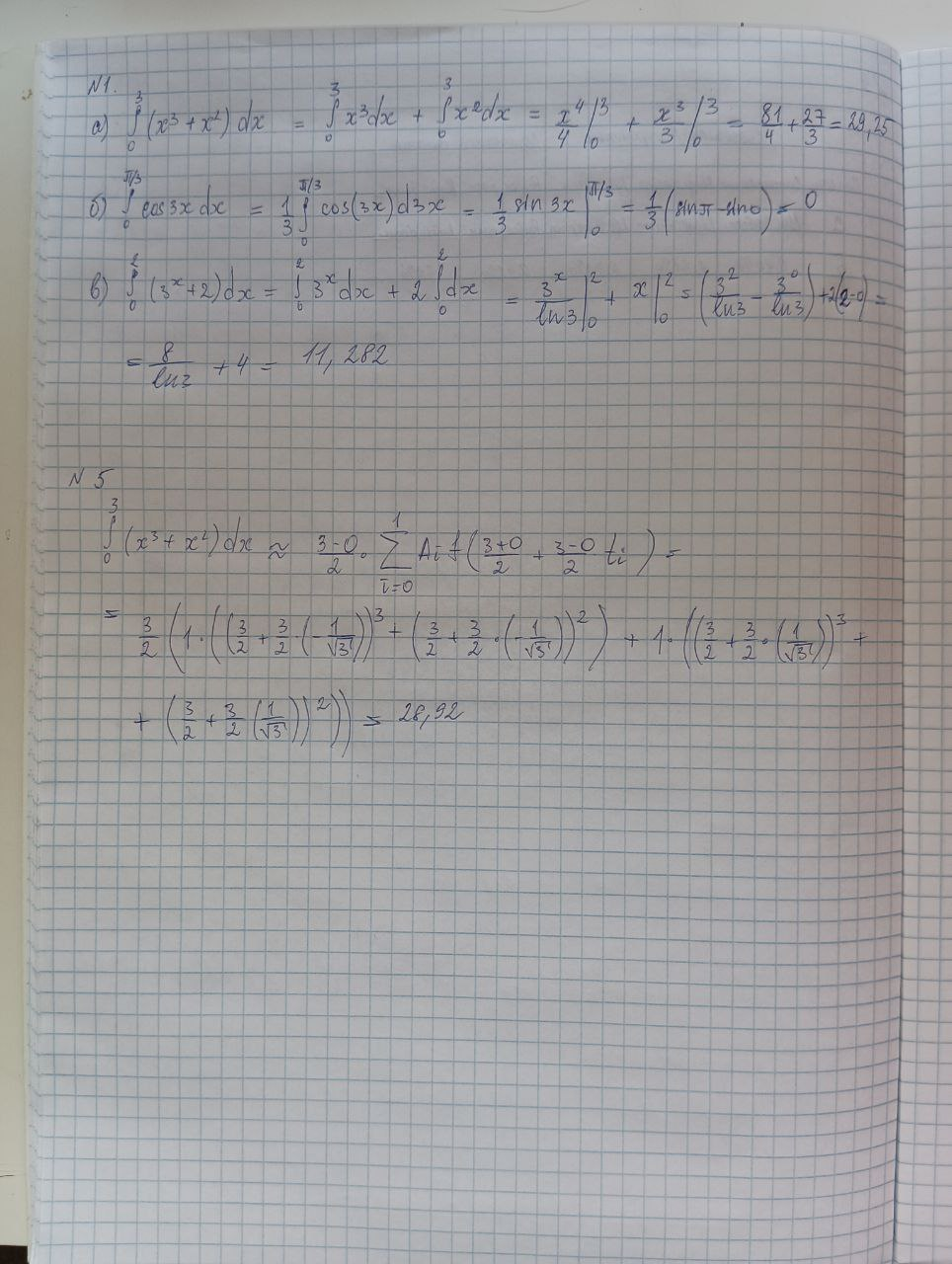
**Цель работы:** изучить понятие квадратурной формулы; изучить основные способы численного интегрирования; получить практические навыки решения задачи численного интегрирования с помощью ЭВМ.

**Ход работы**



1. Вычислить «вручную» интегралы из таблицы вариантов заданий:

− точно (все 3 интеграла);



− по формуле центральных (средних) прямоугольников, используя для оценки точности двойной просчёт при n1=8; n2=10 (интеграл 1);



− по формуле трапеций при n=8 (интеграл 1 и 2);



− по формуле парабол (Симпсона) при n=8 (интеграл 1 и 3).



*Замечание. Для вычисления значений «вручную» рекомендуется использовать Microsoft Excel или другую программу.*

2. Определить погрешность вычисления интеграла 1 по каждой из формул. Результаты представить в виде табл. 3.1.



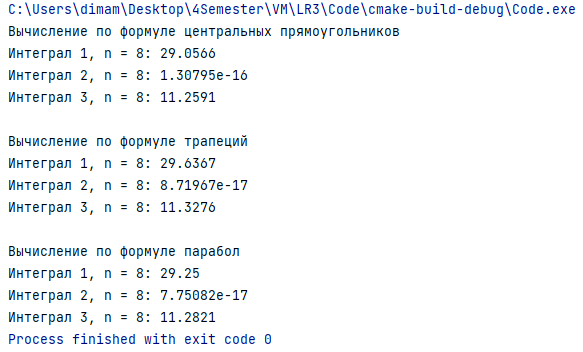
3. Описать в модуле функции, которые возвращают приближенные значения интегралов от функции f(x) с оценкой точности по принципу Рунге для методов центральных прямоугольников, трапеций и парабол.

Исходными данными являются: подынтегральная функция f(x); пределы интегрирования a, b; начальное число отрезков разбиения n; точность вычисления ε. Необходимые для работы значения подынтегральной функции вычисляются непосредственной подстановкой значений аргумента в вычислительную формулу функции.

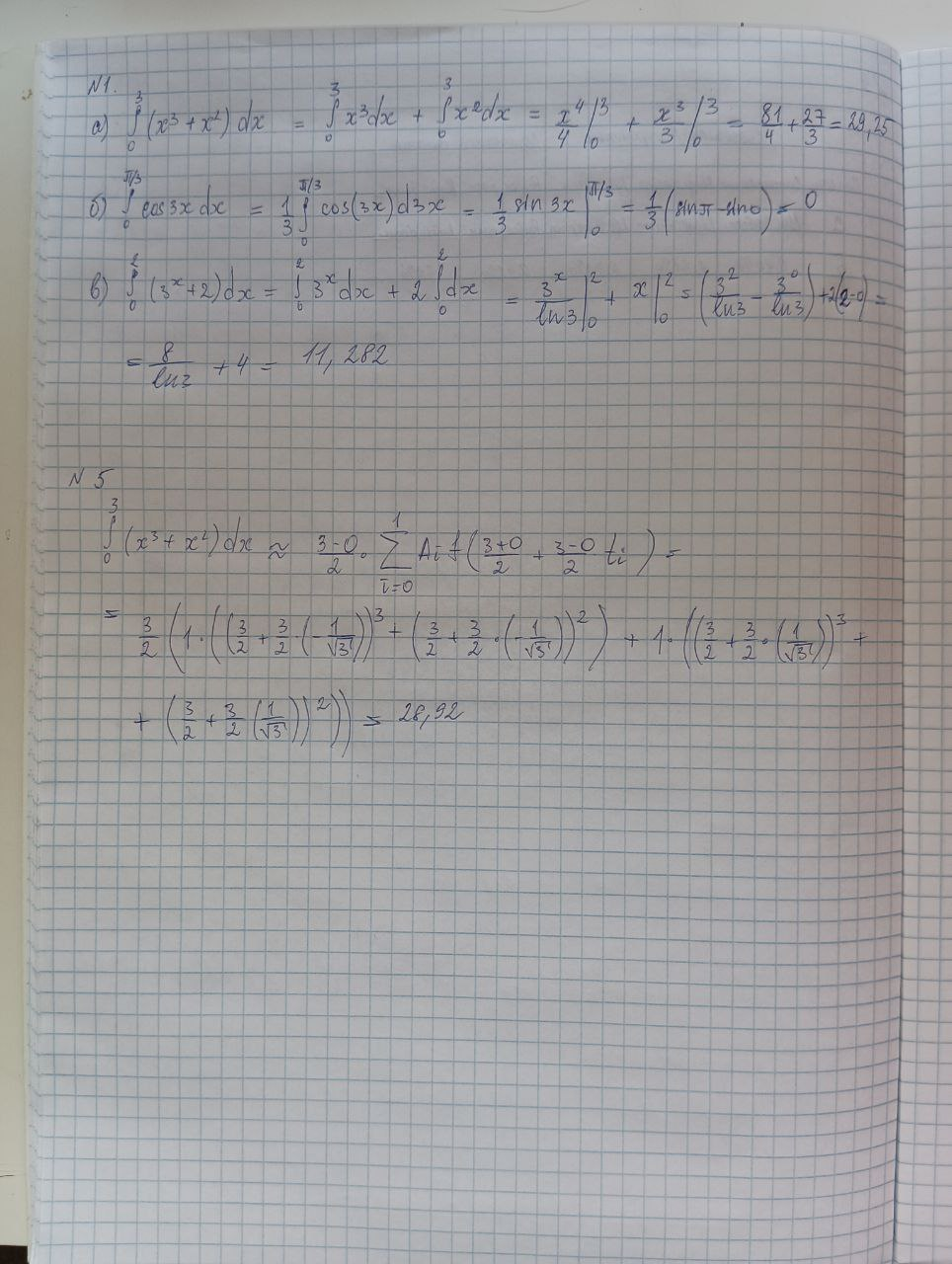
#include "integrals.h"  
#include "utility"  
#include "vector"  
#include "stdlib.h"  
  
  
typedef std::pair<double, double> GridElement;  
  
typedef std::vector<GridElement> Grid;  
  
double calculateIntegralByCentralRectangle(Function function, double lowBorder,  
 double highBorder, int nParts) {  
 Grid grid(nParts);  
  
 double step = (highBorder - lowBorder) / nParts;  
 double sum = 0;  
 for (int i = 0; i < nParts; i++) {  
 grid[i].first = lowBorder + (i + 0.5) \* step;  
 grid[i].second = function(grid[i].first);  
 sum += grid[i].second;  
 }  
  
 return sum \* step;  
}  
  
double calculateIntegralByTrapezoid(Function function, double lowBorder,  
 double highBorder, int nParts) {  
 Grid grid(nParts + 1);  
  
 grid[0] = {lowBorder, function(lowBorder)};  
 grid[grid.size() - 1] = {highBorder, function(highBorder)};  
  
 double step = (highBorder - lowBorder) / nParts;  
 double sum = (grid[0].second + grid[grid.size() - 1].second) / 2;  
 for (int i = 1; i < nParts; i++) {  
 grid[i].first = lowBorder + i \* step;  
 grid[i].second = function(grid[i].first);  
 sum += grid[i].second;  
 }  
  
 return step \* sum;  
}  
  
double calculateIntegralByParable(Function function, double lowBorder,  
 double highBorder, int nParts) {  
 Grid grid(nParts + 1);  
  
 grid[0] = {lowBorder, function(lowBorder)};  
 grid[grid.size() - 1] = {highBorder, function(highBorder)};  
  
 double step = (highBorder - lowBorder) / nParts;  
 double sum = grid[0].second + grid[grid.size() - 1].second;  
  
 for (int i = 1; i < grid.size() - 1; i += 2) {  
 grid[i].first = lowBorder + i \* step;  
 grid[i].second = function(grid[i].first);  
 sum += 4 \* grid[i].second;  
 }  
  
 for (int i = 2; i < grid.size() - 1; i += 2) {  
 grid[i].first = lowBorder + i \* step;  
 grid[i].second = function(grid[i].first);  
 sum += 2 \* grid[i].second;  
 }  
  
 return step / 3 \* sum;  
}  
  
double calculateIntegralByCentralRectangleWithEps(Function function, double lowBorder,  
 double highBorder, int nParts, double &eps) {  
 double integralValueNParts = calculateIntegralByCentralRectangle(function, lowBorder, highBorder, nParts);  
 double difference = abs(integralValueNParts - calculateIntegralByCentralRectangle(function, lowBorder,  
 highBorder, nParts / 2));  
 eps = (double) 1 / 3 \*difference;  
 return integralValueNParts;  
}  
  
double calculateIntegralByTrapezoidWithEps(Function function, double lowBorder,  
 double highBorder, int nParts, double &eps) {  
 double integralValueNParts = calculateIntegralByTrapezoid(function, lowBorder, highBorder, nParts);  
 double difference = abs(integralValueNParts - calculateIntegralByTrapezoid(function, lowBorder,  
 highBorder, nParts / 2));  
 eps = (double) 1 / 3 \* difference;  
 return integralValueNParts;  
}  
  
double calculateIntegralByParableWithEps(Function function, double lowBorder,  
 double highBorder, int nParts, double &eps) {  
 double integralValueNParts = calculateIntegralByParable(function, lowBorder, highBorder, nParts);  
 double difference = abs(integralValueNParts - calculateIntegralByParable(function, lowBorder,  
 highBorder, nParts / 2));  
 eps = (double) 1 / 15 \* difference;  
 return integralValueNParts;  
}

4. Составить программу для вычисления приближенных значений интегралов согласно варианту (все функции из таблицы вариантов заданий) с использованием всех функций, описанных в модуле.

#include <iostream>  
#include "libs/integrals/integrals.h"  
#include "cmath"  
#include "windows.h"  
  
double f1(double x) {  
 return pow(x, 3) + pow(x, 2);  
}  
  
double f2(double x) {  
 return cos(3 \* x);  
}  
  
double f3(double x) {  
 return pow(3, x) + 2;  
}  
  
int main() {  
 SetConsoleOutputCP(**CP\_UTF8**);  
  
 std::cout << "Вычисление по формуле центральных прямоугольников\n" <<  
 "Интеграл 1, n = 8: " << calculateIntegralByCentralRectangle(f1, 0, 3, 8) <<  
 "\nИнтеграл 1, n = 10: " << calculateIntegralByCentralRectangle(f1, 0, 3, 10) <<  
 "\n\nВычисление по формуле трапеции\n" <<  
 "Интеграл 1, n = 8: " << calculateIntegralByTrapezoid(f1, 0, 3, 8) <<  
 "\nИнтеграл 2, n = 8: " << calculateIntegralByTrapezoid(f2, 0, std::numbers::pi / 3, 8) <<  
 "\n\nВычисление по формуле парабол\n" <<  
 "Интеграл 1, n = 8: " << calculateIntegralByParable(f1, 0, 3, 8) <<  
 "\nИнтеграл 3, n = 8: " << calculateIntegralByParable(f3, 0, 3, 8);  
}



5. Вычислить «вручную» интеграл из столбца 1 табл. 3.1 по формуле Гаусса при n = 2. Значения узлов ti и весов Аi приведены в табл. 3.2.

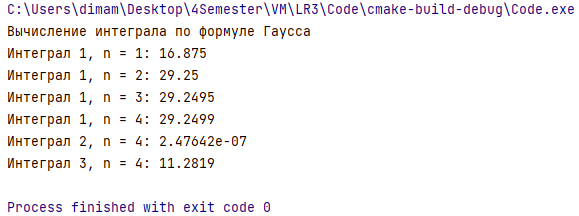


6. Описать в модуле функцию для вычисления приближенного значения интеграла от функции f(x) по формуле Гаусса при n = 1, 2, 3, 4. Добавить в составленную программу вычисление приближенных значений интегралов (все функции из таблицы вариантов заданий) с использованием формулы Гаусса.

struct gaussCoefficient {  
 double t;  
 double A;  
};

std::vector<std::vector<gaussCoefficient>> gaussCoefficients**{** {},  
 {{0, 2}},  
 {{-0.57735, 1}, {0.57735, 1}},  
 {{-0.77459, 0.55555}, {0, 0.88888}, {0.77459, 0.55555}},  
 {{-0.86114, 0.34785}, {-0.33998, 0.65215}, {0.33998, 0.65215}, {0.861136, 0.34785}}  
**}**;  
  
double calculateIntegralByGauss(Function function, double lowBorder,  
 double highBorder, int nParts) {  
 double sum = 0;  
 for (int i = 0; i < nParts; i++) {  
 double value = gaussCoefficients[nParts][i].A;  
 value \*= function((highBorder + lowBorder) / 2 +  
 (highBorder - lowBorder) / 2 \* gaussCoefficients[nParts][i].t);  
 sum += value;  
 }  
  
 return (highBorder - lowBorder) / 2 \* sum;  
}

int main() {  
 SetConsoleOutputCP(**CP\_UTF8**);  
  
 std::cout << "Вычисление интеграла по формуле Гаусса\n";  
  
 for (int i = 1; i <= 4; i++) {  
 std::cout << "Интеграл 1, n = " << i << ": " <<  
 calculateIntegralByGauss(f1, 0, 3, i) << "\n";  
 }  
  
 std::cout << "Интеграл 2, n = 4: " <<  
 calculateIntegralByGauss(f2, 0, std::numbers::pi / 3, 4) << "\n";  
 std::cout << "Интеграл 3, n = 4: " <<  
 calculateIntegralByGauss(f3, 0, 2, 4) << "\n";  
}



7. Заполнить значения погрешности вычисления интеграла 1 в таблице 3.1 для формулы Гаусса при n = 1, 2, 3, 4.



**Вывод:** в ходе лабораторной работы мы изучили понятие квадратурной формулы; изучили основные способы численного интегрирования; получили практические навыки решения задачи численного интегрирования с помощью ЭВМ.