Задача 12. Функции

1. Описать функцию Mean, вычисляющую среднее арифметическое и среднее геометрическое двух положительных чисел

#include <cmath>

// Функция для вычисления среднего арифметического и среднего геометрического

void Mean(double a, double b, double& arithmeticMean, double& geometricMean)

{

arithmeticMean = (a + b) / 2.0; // вычисляем среднее арифметическое

geometricMean = std::sqrt(a \* b); // вычисляем среднее геометрическое

}

// Пример использования функции

int main()

{

double a = 4.0;

double b = 9.0;

double arithmeticMean, geometricMean;

// вызов функции Mean

Mean(a, b, arithmeticMean, geometricMean);

// вывод результата

std::cout << "Среднее арифметическое: " << arithmeticMean << std::endl;

std::cout << "Среднее геометрическое: " << geometricMean << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

2. Описать функцию CircleS, находящую площадь круга радиуса R.

#include <cmath>

// Функция для вычисления площади круга радиуса R

double CircleS(double R)

{

double S = M\_PI \* R \* R; // вычисляем площадь круга

return S;

}

// Пример использования функции

int main()

{

double R = 5.0;

double S = CircleS(R);

// вывод результата

std::cout << "Площадь круга радиуса " << R << " равна " << S << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

3. Описать функцию TrianglePS, вычисляющую по стороне a равностороннего треугольника его периметр P = 3·a и площадь S = a2 3/4.

#include <cmath>

// Функция для вычисления периметра и площади равностороннего треугольника по стороне a

void TrianglePS(double a, double& P, double& S)

{

P = 3.0 \* a; // вычисляем периметр

S = (a \* a \* std::sqrt(3)) / 4.0; // вычисляем площадь

}

// Пример использования функции

int main()

{

double a = 5.0;

double P, S;

// вызов функции TrianglePS

TrianglePS(a, P, S);

// вывод результата

std::cout << "Периметр треугольника: " << P << std::endl;

std::cout << "Площадь треугольника: " << S << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

4. Описать функцию RingS, находящую площадь кольца, заключенного между двумя окружно- стями с общим центром и радиусами R1 и R2.

#include <cmath>

// Функция для вычисления площади кольца между двумя окружностями

double RingS(double R1, double R2)

{

double S1 = M\_PI \* R1 \* R1; // площадь большей окружности

double S2 = M\_PI \* R2 \* R2; // площадь меньшей окружности

double S = S1 - S2; // площадь кольца между окружностями

return S;

}

// Пример использования функции

int main()

{

double R1 = 10.0;

double R2 = 5.0;

double S = RingS(R1, R2);

// вывод результата

std::cout << "Площадь кольца между окружностями радиусов " << R1 << " и " << R2 << " равна " << S << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

5. Описать функцию RectPS, вычисляющую периметр P и площадь S прямоугольника со сторо- нами, параллельными осям координат, по координатам (x1, y1), (x2, y2) его противоположных вершин.

// Функция для вычисления периметра и площади прямоугольника по его координатам

void RectPS(double x1, double y1, double x2, double y2, double &P, double &S)

{

double a = fabs(x2 - x1); // длина одной из сторон прямоугольника

double b = fabs(y2 - y1); // длина другой стороны прямоугольника

P = 2 \* (a + b); // периметр прямоугольника

S = a \* b; // площадь прямоугольника

}

// Пример использования функции

int main()

{

double x1 = 0.0;

double y1 = 0.0;

double x2 = 5.0;

double y2 = 3.0;

double P, S;

RectPS(x1, y1, x2, y2, P, S);

// вывод результата

std::cout << "Периметр прямоугольника: " << P << std::endl;

std::cout << "Площадь прямоугольника: " << S << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

6. Описать функцию TriangleP, находящую периметр равнобедренного треугольника по его ос- нованию a и высоте h, проведенной к основанию. Для нахождения боковой стороны b тре- угольника использовать теорему Пифагора: b2 =(a/2)2 +h2.

// Функция для вычисления периметра равнобедренного треугольника по его основанию и высоте

double TriangleP(double a, double h)

{

double b = sqrt((a\*a)/4 + h\*h); // длина боковой стороны треугольника

double P = 2\*b + a; // периметр треугольника

return P;

}

// Пример использования функции

int main()

{

double a = 5.0;

double h = 3.0;

double P = TriangleP(a, h);

// вывод результата

std::cout << "Периметр равнобедренного треугольника: " << P << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

7. Описать функцию InvertDigits, меняющую порядок следования цифр целого положительного числа K на обратный.

// Функция для инвертирования цифр числа

int InvertDigits(int K)

{

int reversedNumber = 0;

while (K != 0)

{

reversedNumber = reversedNumber \* 10 + K % 10;

K /= 10;

}

return reversedNumber;

}

// Пример использования функции

int main()

{

int K = 12345;

int reversedNumber = InvertDigits(K);

// вывод результата

std::cout << "Число с инвертированным порядком цифр: " << reversedNumber << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

8. Описать функцию SumRange, находящую сумму всех целых чисел от A до B включительно. Если A > B, то функция возвращает 0.

// Функция для нахождения суммы всех целых чисел от A до B включительно

int SumRange(int A, int B)

{

if (A > B) {

return 0;

}

int sum = 0;

for (int i = A; i <= B; i++) {

sum += i;

}

return sum;

}

// Пример использования функции

int main()

{

int A = 1;

int B = 10;

int sum = SumRange(A, B);

// вывод результата

std::cout << "Сумма всех целых чисел от " << A << " до " << B << " равна: " << sum << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

9. Определить функцию PTrap для расчета периметра равнобедренной трапеции по ее основа- нию и высоте. Задать основания и высоты двух равнобедренны трапеций. Найти сумму их периметров.

// Функция для расчета периметра равнобедренной трапеции по ее основанию и высоте

double PTrap(double a, double h)

{

double b = sqrt(pow(a / 2, 2) + pow(h, 2)); // находим длину боковой стороны

return a + 2 \* b; // вычисляем периметр

}

// Пример использования функции

int main()

{

double a1 = 10; // основание первой трапеции

double h1 = 5; // высота первой трапеции

double a2 = 15; // основание второй трапеции

double h2 = 7; // высота второй трапеции

double p1 = PTrap(a1, h1); // находим периметр первой трапеции

double p2 = PTrap(a2, h2); // находим периметр второй трапеции

double sum\_p = p1 + p2; // находим сумму периметров

// вывод результата

std::cout << "Периметр первой трапеции: " << p1 << std::endl;

std::cout << "Периметр второй трапеции: " << p2 << std::endl;

std::cout << "Сумма периметров: " << sum\_p << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

10. Описать функцию Quarter, определяющую номер координатной четверти, в которой находит- ся точка с ненулевыми вещественными координатами (x, y).

int Quarter(double x, double y) {

if (x > 0 && y > 0) {

// Точка находится в первой четверти

return 1;

} else if (x < 0 && y > 0) {

// Точка находится во второй четверти

return 2;

} else if (x < 0 && y < 0) {

// Точка находится в третьей четверти

return 3;

} else if (x > 0 && y < 0) {

// Точка находится в четвертой четверти

return 4;

} else {

// Точка находится на оси координат или совпадает с началом координат

return 0;

}

}

////////////////////////////////////////////////////////////////////////////

11. Описать функцию DigitCountSum, находящую количество цифр целого положительного чис- ла, а также их сумму.

void DigitCountSum(int num, int& count, int& sum) {

count = 0;

sum = 0;

while(num > 0) {

int digit = num % 10; // извлекаем последнюю цифру

count++; // увеличиваем количество цифр

sum += digit; // добавляем цифру к сумме

num /= 10; // убираем последнюю цифру из числа

}

}

////////////////////////////////////////////////////////////////////////////

12. Описать функцию Calc, выполняющую над ненулевыми вещественными числами A и B одну из арифметических операций и возвращающую ее результат. Вид операции определяется це- лым параметром: 1 – вычитание, 2 – умножение, 3 – деление, остальные значения – сложение.

double Calc(double A, double B, int operation) {

double result;

switch(operation) {

case 1:

result = A - B;

break;

case 2:

result = A \* B;

break;

case 3:

result = A / B;

break;

default:

result = A + B;

}

return result;

}

////////////////////////////////////////////////////////////////////////////

13. Написать программу определения периметра треугольника, заданного координатами его вершин. Длину стороны определять с помощью функции.

#include <iostream>

#include <cmath>

using namespace std;

// функция для вычисления длины стороны треугольника по координатам его вершин

double getLength(double x1, double y1, double x2, double y2) {

double d = sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));

return d;

}

int main() {

double x1, y1, x2, y2, x3, y3;

double a, b, c, p, perimeter;

// ввод координат вершин

cout << "Enter coordinates of triangle vertices:" << endl;

cout << "Vertex 1 (x, y): ";

cin >> x1 >> y1;

cout << "Vertex 2 (x, y): ";

cin >> x2 >> y2;

cout << "Vertex 3 (x, y): ";

cin >> x3 >> y3;

// вычисление длин сторон треугольника

a = getLength(x1, y1, x2, y2);

b = getLength(x2, y2, x3, y3);

c = getLength(x3, y3, x1, y1);

// вычисление периметра

perimeter = a + b + c;

// вывод результата

cout << "Perimeter of the triangle is: " << perimeter << endl;

return 0;

}

Enter coordinates of triangle vertices:

Vertex 1 (x, y): 0 0

Vertex 2 (x, y): 4 0

Vertex 3 (x, y): 0 3

Perimeter of the triangle is: 12

////////////////////////////////////////////////////////////////////////////

14. Описать функцию DegToRad, находящую величину угла в радианах, если дана его величина D в градусах (D – вещественное число, 0 < D < 360). Воспользоваться следующим соотноше- нием: 180° = π радианов.

#include <cmath>

double DegToRad(double degree) {

const double PI = 3.14159265358979323846;

return degree \* PI / 180.0;

}

////////////////////////////////////////////////////////////////////////////

15. Описать функцию IsLeapYear, определяющую, является заданный год високосным. Високос- ным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400.

bool IsLeapYear(int year) {

if (year % 4 == 0) {

if (year % 100 == 0) {

return year % 400 == 0;

} else {

return true;

}

} else {

return false;

}

}

////////////////////////////////////////////////////////////////////////////

Задача 13. Рекурсия

1. Описать рекурсивные функции Fact(N) и Fact2(N) вещественного типа, вычисляющие значе- ния факториала N! и двойного факториала N!! соответственно (N > 0 — параметр целого ти- па).

// Функция вычисления факториала N!

double Fact(int N) {

if (N == 0) { // базовый случай

return 1;

} else {

return N \* Fact(N - 1); // рекурсивный случай

}

}

// Функция вычисления двойного факториала N!!

double Fact2(int N) {

if (N == 0 || N == 1) { // базовый случай

return 1;

} else {

return N \* Fact2(N - 2); // рекурсивный случай

}

}

////////////////////////////////////////////////////////////////////////////

2. Описать рекурсивную функцию PowerN(x, n) вещественного типа, находящую значение n-й степеничислаxпоформуле:x0 =1,xn =x·xn–1 приn>0,xn =1/x–n приn<0(x>=0–вещест- венное число, n – целое).

#include <iostream>

using namespace std;

double PowerN(double x, int n) {

if (n == 0) {

return 1;

} else if (n > 0) {

return x \* PowerN(x, n - 1);

} else {

return 1 / PowerN(x, -n);

}

}

int main() {

double x;

int n;

cout << "Enter x: ";

cin >> x;

cout << "Enter n: ";

cin >> n;

cout << "Result: " << PowerN(x, n) << endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

3. Описать рекурсивную функцию SqrtK(x, k, n) вещественного типа, находящую приближенное значение корня k-й степени из числа x по формуле: y(0)=1, y(n+1)= y(n)–(y(n)–x/y (n))/k, где y(n) обозначает SqrtK(x, k, n) (x – вещественный параметр, k и n – целые; x > 0, k > 1, n > 0).

double SqrtK(double x, int k, int n) {

if (n == 0) {

return 1.0;

} else {

double y = SqrtK(x, k, n-1);

return y - (y - x / pow(y, k-1)) / k;

}

}

////////////////////////////////////////////////////////////////////////////

4. Описать рекурсивную функцию FibRec(N) целого типа, вычисляющую N-е число Фибоначчи F(N) по формуле: F(1) = F(2) = 1, F(k) = F(k–2) + F(k–1), k = 3, 4, ... С помощью этой функции найти пять чисел Фибоначчи с указанными номерами и вывести эти числа вместе с количест- вом рекурсивных вызовов функции FibRec, потребовавшихся для их нахождения.

int FibRec(int N, int& calls) {

calls++;

if (N == 1 || N == 2) {

return 1;

}

return FibRec(N - 2, calls) + FibRec(N - 1, calls);

}

int main() {

int calls = 0;

for (int i = 1; i <= 5; i++) {

int fib = FibRec(i, calls);

cout << "Fibonacci(" << i << ") = " << fib << ", calls = " << calls << endl;

calls = 0;

}

return 0;

}

////////////////////////////////////////////////////////////////////////////

5. Описать рекурсивную функцию C(m, n) целого типа, находящую число сочетаний из n эле- ментов по m, используя формулу: C(0, n) = C(n, n) = 1, C(m, n) = C(m, n–1) + C(m–1, n–1) при 0 < m < n (m и n — целые параметры; n > 0, 0 <= m <= n). Дано число N и пять различных зна- чений M. Вывести числа C(M, N) вместе с количеством рекурсивных вызовов функции C, по- требовавшихся для их нахождения.

int C(int m, int n) {

if (m == 0 || m == n) {

return 1;

} else {

return C(m, n-1) + C(m-1, n-1);

}

}

#include <iostream>

using namespace std;

int C(int m, int n, int& count) {

count++; // увеличиваем счетчик рекурсивных вызовов

if (m == 0 || m == n) {

return 1;

} else {

return C(m, n-1, count) + C(m-1, n-1, count);

}

}

int main() {

int N = 5;

int M[] = {0, 1, 2, 3, 4, 5};

int num\_M = sizeof(M) / sizeof(M[0]);

for (int i = 0; i < num\_M; i++) {

int count = 0;

int result = C(M[i], N, count);

cout << "C(" << M[i] << ", " << N << ") = " << result << ", количество вызовов: " << count << endl;

}

return 0;

}

////////////////////////////////////////////////////////////////////////////

6. Описать рекурсивную функцию NOD(A, B) целого типа, находящую наибольший общий де- литель двух натуральных чисел A и B, используя алгоритм Евклида: NOD(A, B) = NOD(B mod A, A), если A <> 0; NOD(0, B) = B. С помощью этой функции найти наибольшие общие делители пар A и B, A и C, A и D, если даны числа A, B, C, D.

int NOD(int A, int B) {

if (A == 0) {

return B;

}

return NOD(B % A, A);

}

int main() {

int A = 24, B = 36, C = 45, D = 63;

std::cout << "NOD(A, B) = " << NOD(A, B) << std::endl;

std::cout << "NOD(A, C) = " << NOD(A, C) << std::endl;

std::cout << "NOD(A, D) = " << NOD(A, D) << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

7. Описать рекурсивную функцию MinRec(A, N) вещественного типа, которая находит мини- мальный элемент вещественного массива A размера N, не используя оператор цикла. С по- мощью функции MinRec найти минимальные элементы массивов A, B, C размера NA, NB, NC соответственно.

#include <iostream>

using namespace std;

double MinRec(double A[], int N) {

if (N == 1) {

return A[0];

} else {

double min\_of\_rest = MinRec(A+1, N-1);

if (A[0] < min\_of\_rest) {

return A[0];

} else {

return min\_of\_rest;

}

}

}

int main() {

const int NA = 5, NB = 6, NC = 7;

double A[NA] = {1.1, -2.2, 3.3, 4.4, -5.5};

double B[NB] = {0.0, 1.0, 2.0, 3.0, -4.0, -5.0};

double C[NC] = {5.5, 4.4, 3.3, 2.2, 1.1, 0.0, -1.1};

cout << "Min of A: " << MinRec(A, NA) << endl;

cout << "Min of B: " << MinRec(B, NB) << endl;

cout << "Min of C: " << MinRec(C, NC) << endl;

return 0;

}

Min of A: -5.5

Min of B: -5

Min of C: -1.1

////////////////////////////////////////////////////////////////////////////

8. Описать рекурсивную функцию Digits(S) целого типа, находящую количество цифр в строке S без использования оператора цикла. С помощью этой функции найти количество цифр в заданной строке.

int Digits(const std::string& s)

{

if (s.empty()) {

return 0; // строка пустая, цифр нет

}

if (isdigit(s[0])) {

return 1 + Digits(s.substr(1)); // первый символ - цифра, переходим к следующей части строки

}

return Digits(s.substr(1)); // первый символ не цифра, переходим к следующей части строки

}

#include <iostream>

#include <string>

int main()

{

std::string s = "abc123def456";

int digits = Digits(s);

std::cout << "Количество цифр в строке " << s << " равно " << digits << std::endl;

return 0;

}

Количество цифр в строке abc123def456 равно 6

////////////////////////////////////////////////////////////////////////////

9. Описать рекурсивную функцию Simm(S) логического типа, проверяющую, является ли сим- метричной строка S, без использования оператора цикла. С помощью этой функции прове- рить заданную строку.

#include <iostream>

#include <string>

using namespace std;

bool Simm(string s, int left, int right) {

if (left >= right) { // базовый случай - строка симметрична

return true;

}

if (s[left] != s[right]) { // базовый случай - строка не симметрична

return false;

}

// рекурсивный случай - сравниваем символы с обеих сторон

return Simm(s, left + 1, right - 1);

}

int main() {

string s;

cout << "Введите строку: ";

getline(cin, s);

if (Simm(s, 0, s.length() - 1)) {

cout << "Строка симметрична" << endl;

} else {

cout << "Строка не симметрична" << endl;

}

return 0;

}

////////////////////////////////////////////////////////////////////////////

10. Задано положительное и отрицательное число в двоичной системе. Составить программу вы- числения суммы этих чисел, используя функцию сложения чисел в двоичной системе счис- ления.

#include <iostream>

#include <string>

using namespace std;

string addBinary(string a, string b) {

string result = ""; // строка для записи результата

int carry = 0; // запоминаем единицу, если она появилась

int i = a.length() - 1; // индекс последнего символа в первом числе

int j = b.length() - 1; // индекс последнего символа во втором числе

// пока не дошли до начала обоих чисел

while (i >= 0 || j >= 0) {

int sum = carry; // начинаем с единицы, если она была запомнена

if (i >= 0) {

sum += a[i] - '0'; // прибавляем очередной символ из первого числа

i--;

}

if (j >= 0) {

sum += b[j] - '0'; // прибавляем очередной символ из второго числа

j--;

}

result = to\_string(sum % 2) + result; // записываем очередной разряд суммы в начало результата

carry = sum / 2; // запоминаем единицу, если она появилась

}

if (carry != 0) {

result = to\_string(carry) + result; // добавляем единицу в начало результата

}

return result;

}

////////////////////////////////////////////////////////////////////////////

11. Описать рекурсивную функцию Root (а, b, ε), которая методом деления отрезка пополам на- ходит с точностью ε корень уравнения f(x) = 0 на отрезке [а, b] (считать, что ε > 0, а < b, f(a) – f(b) < 0 и f(x) — непрерывная и монотонная на отрезке [а, b] функция).

double Root(double a, double b, double eps, double (\*f)(double)) {

double m = (a + b) / 2; // находим середину отрезка

if (b - a < eps) { // если длина отрезка стала меньше заданной точности, возвращаем середину

return m;

}

if (f(a) \* f(m) < 0) { // если знак функции на концах отрезка разный, ищем корень на левой половине

return Root(a, m, eps, f);

} else { // иначе ищем корень на правой половине

return Root(m, b, eps, f);

}

}

#include <iostream>

#include <cmath>

using namespace std;

double f(double x) {

return x \* x - 2;

}

int main() {

double a = 1, b = 2, eps = 0.0001;

double root = Root(a, b, eps, f);

cout << "Root of x^2 - 2 on [" << a << ", " << b << "] with eps = " << eps << " is " << root << endl;

return 0;

}

Root of x^2 - 2 on [1, 2] with eps = 0.0001 is 1.41421

////////////////////////////////////////////////////////////////////////////

12. Описать функцию min(X) для определения минимального элемента линейного массива X, введя вспомогательную рекурсивную функцию minl(k), находящую минимум среди послед- них элементов массива X, начиная с k-го.

int minl(int X[], int n, int k) {

if (k == n - 1) { // базовый случай - один элемент

return X[k];

} else {

int m = minl(X, n, k + 1); // находим минимум среди последующих элементов

if (X[k] < m) {

return X[k]; // если текущий элемент меньше, то возвращаем его

} else {

return m; // иначе возвращаем минимум из последующих

}

}

}

int min(int X[], int n) {

return minl(X, n, 0); // начинаем поиск минимума с первого элемента

}

int X[] = { 5, 3, 8, 1, 6 };

int n = sizeof(X) / sizeof(X[0]);

int min\_elem = min(X, n);

cout << "Минимальный элемент: " << min\_elem << endl; // Минимальный элемент: 1

////////////////////////////////////////////////////////////////////////////

13. Описать рекурсивную логическую функцию Simm(S, I, J), проверяющую, является ли сим- метричной часть строки S, начинающаяся i-м и заканчивающаяся j-м ее элементами.

bool Simm(string S, int I, int J) {

if (I >= J) {

return true; // строка пустая или содержит один символ

}

if (S[I] != S[J]) {

return false; // символы на концах различны

}

return Simm(S, I + 1, J - 1); // проверяем симметрию внутренней части

}

////////////////////////////////////////////////////////////////////////////

14. Составить программу для нахождения числа, которое образуется из данного натурального числа при записи его цифр в обратном порядке. Например, для числа 1234 получаем резуль- тат 4321.

#include <iostream>

int reverse(int num, int rev = 0) {

if (num == 0) {

return rev;

} else {

rev = rev \* 10 + num % 10;

return reverse(num / 10, rev);

}

}

int main() {

int num;

std::cout << "Enter a natural number: ";

std::cin >> num;

int rev = reverse(num);

std::cout << "The reverse of " << num << " is " << rev << std::endl;

return 0;

}

////////////////////////////////////////////////////////////////////////////

15. Составить программу для перевода данного натурального числа в р-ичную систему счисле- ния (2 ≤ р ≤ 9).

#include <iostream>

using namespace std;

// Функция перевода числа num в систему с основанием p

void convert(int num, int p) {

if (num == 0) {

return;

}

convert(num / p, p);

cout << num % p;

}

int main() {

int num, p;

cout << "Введите число: ";

cin >> num;

cout << "Введите основание системы счисления (2 <= p <= 9): ";

cin >> p;

cout << "Число " << num << " в " << p << "-ичной системе счисления: ";

convert(num, p);

cout << endl;

return 0;

}