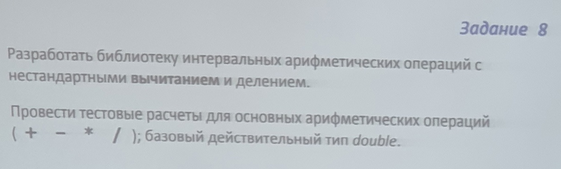
Код программы  
#include <iostream>



#include <cmath>

#include <iomanip>

#include <algorithm>

#include <fenv.h>

#include <cstring>

using namespace std;

class Interval {

public:

double L, R;

Interval(double left, double right) : L(left), R(right) {}

// Перегрузка оператора сложения для интервалов

Interval operator+(const Interval& other) const {

Interval result(L,R);

result.L = this->L + other.L;

result.R = this->R + other.R;

return result;

}

// Перегрузка оператора умножения для интервалов

Interval operator\*(const Interval& other) const {

Interval result(L,R);

double a = this->L;

double b = this->R;

double c = other.L;

double d = other.R;

result.L = min(min(a \* c, a \* d), min(b \* c, b \* d));

result.R = max(max(a \* c, a \* d), max(b \* c, b \* d));

return result;

}

// Перегрузка оператора вычитания для интервалов

// НЕСТАНДАРТНАЯ ОПЕРАЦИЯ ВЫЧИТАНИЯ

Interval operator-(const Interval& other) const {

Interval result(L, R);

double a1 = this->L;

double a2 = this->R;

double b1 = other.L;

double b2 = other.R;

result.L = min(a1 - b1, a2 - b2);

result.R = max(a1 - b1, a2 - b2);

return result;

}

// Перегрузка оператора деления для интервалов

// НЕСТАНДАРТНАЯ ОПЕРАЦИЯ ДЕЛЕНИl

Interval operator/(const Interval& other) const {

if (\*this \* other > Interval(0, 0)) {

double tmp[] = {L / other.L, R / other.R};

std::sort(tmp, tmp + 2);

return {tmp[0], tmp[1]};

} else if (\*this > Interval(0, 0) && other > Interval(0, 0)) {

double tmp[] = {L / other.R, R / other.L};

std::sort(tmp, tmp + 2);

return {tmp[0], tmp[1]};

} else if (L <= 0 && R >= 0 && other.L > 0 && other.R > 0) {

double tmp = 1 / other.L;

return {tmp \* L, tmp \* R};

} else if (L <= 0 && R >= 0 && other.L < 0 && other.R < 0) {

double tmp = 1 / other.R;

return {tmp \* L, tmp \* R};

} else {

throw std::runtime\_error("Данные вычисления не предусмотрены библиотекой");

}

}

// Перегрузка оператора меньше (<)

bool operator<(const Interval& other) const {

return this->R < other.L;

}

// Перегрузка оператора больше (>)

bool operator>(const Interval& other) const {

return this->L > other.R;

}

// Перегрузка оператора сравнения равенства для интервалов

bool operator==(const Interval& other) const {

return this->L == other.L && this->R == other.R;

}

// Перегрузка оператора сравнения неравенства для интервалов

bool operator!=(const Interval& other) const {

return this->L != other.L || this->R != other.R;

}

};

void testOperation(const char\* operation, Interval& a, const Interval& b) {

printf("\n\t\tТестирование операции %s\n", operation);

printf(" N\t\t\tleft\t\t\tright\t\t\tmed\t\t wid\n");

int count = 1;

for (int i = 1; i <= 1000000000; ++i) {

if (std::strcmp(operation, "сложения") == 0)

a = a + b;

else if (std::strcmp(operation, "вычитания") == 0)

a = a - b;

else if (std::strcmp(operation, "умножения") == 0)

a = a \* b;

else if (std::strcmp(operation, "деления") == 0)

a = a / b;

if (i == 200000000 \* count) {

printf("%d\t%.15f\t%.15f\t%.15f\t%.2e\n", i, a.L, a.R, (a.L + a.R) / 2, (a.L + a.R) / 2 - a.L);

count++;

}

}

printf("\n\n");

}

int main() {

Interval a(0.0, 0.0);

Interval b(0.000000001, 0.000000001);

testOperation("сложения", a, b);

a.L = 1.0;

a.R = 1.0;

b.L = 0.000000001;

b.R = 0.000000001;

testOperation("вычитания", a, b);

a.L = 1.0;

a.R = 1.0;

b.L = 1.000000001;

b.R = 1.000000001;

testOperation("умножения", a, b);

a.L = 1.0;

a.R = 1.0;

b.L = 1.000000001;

b.R = 1.000000001;

testOperation("деления", a, b);

return 0;

}

