**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

Практические приемы построения многопоточных приложений. Вариант 18

**Исполнитель**

Студент группы БПИ193

Радайкин В. А.

**Задание**

Задача о наследстве. У старого дона Энрике было два сына, у каждого из сыновей – еще по два сына, каждый из которых имел еще по два сына. Умирая, дон Энрике завещал все свое богатство правнукам в разных долях. Адвокат дон Хосе выполнил задачу дележа наследства в меру своих способностей. Правнуки заподозрили адвоката в укрывательстве части наследства. Требуется создать многопоточное приложение, которое при известных сумме завещания дона Энрике и доле каждого наследника, проверяет честность адвоката. При решении использовать принцип дихотомии.

Составление​​ программы.

С помощью принципа дихотомии параллельно суммируем доли двух сыновей одного отца, затем поднимаемся «вверх» к предкам, суммируя доли сыновей. В программе используются потоки из стандартной библиотеки C++.

Текст программы:

// Радайкин Владислав БПИ193. Вариант 18

#include <iostream>

#include <string>

#include <vector>

#include <thread>

#include <cmath>

using namespace std;

// Метод для считывания числа

double readNum(int maxBound) {

double num = -1;

string str;

do {

int isCorrect = 0;

while (!isCorrect) {

isCorrect++;

cout << "Введите положительное число до " << maxBound << endl;

cin >> str;

try {

num = stod(str);

}

catch (const std::exception&) {

isCorrect--;

}

}

} while (num <= 0 || num > maxBound);

return num;

}

void addPart(double part, double& num) {

num += part;

}

void toYoung(double part1, double part2, double& num) {

thread\* threads[2];

threads[0] = new thread{ addPart, part1, ref(num) };

threads[1] = new thread{ addPart, part2, ref(num) };

for (int i = 0; i < 2; i++) {

threads[i]->join();

delete threads[i];

}

}

void toOld(double part1, double part2, double part3, double part4, double& num) {

thread\* threads[2];

threads[0] = new thread{ toYoung, part1, part2, ref(num) };

threads[1] = new thread{ toYoung, part3, part4, ref(num) };

for (int i = 0; i < 2; i++) {

threads[i]->join();

delete threads[i];

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

vector<double> parts; // Вектор для хранения частей наследства

double gotPart = 0; // Число для хранения имеющейся доли наследства

cout << "Общая сумма наследства" << endl;

double numAll = readNum(INT\_MAX);

// Получение частей наследства

for (int i = 0; i < 8; i++) {

cout << "Процент от общей суммы " << i + 1 << "-ого сына" << endl;

double part = readNum(100) / 100;

parts.push\_back(part);

}

// Считаем полученную внуками долю

thread\* threads[2];

for (int i = 0; i < 2; i++) {

threads[i] = new thread{ toOld, parts[i \* 4], parts[i \* 4 + 1], parts[i \* 4 + 2], parts[i \* 4 + 3], ref(gotPart) };

}

// Ждем окончания работы потоков

for (int i = 0; i < 2; i++) {

threads[i]->join();

delete threads[i];

}

double realNum = gotPart \* numAll;

if (realNum == numAll) {

cout << "Наследство распределено верно" << endl;

}

else {

cout << "Адвокат ошибся на " << abs(realNum - numAll) << endl;

}

return 0;

}