МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

по дисциплине
«Структуры и алгоритмы обработки данных»

Тема. Алгоритмические стратегии. Разработка и программная реализация задач с применением метода сокращения числа переборов

Выполнил студент группы ИКБО-25-22 Ракитин В.А.

Принял доцент Бузыкова Ю.С.

Москва 2023

Задание:

№	Задача	Метод
11	Разработать программу расстановки на 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого».	метод ветвей и границ

Ход работы:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <chrono>

const int N = 8; // Размер доски

bool isSafe(const std::vector<int>& placement, int row, int col) {
    for (int i = 0; i < row; ++i) {
        if (placement[i] == col || std::abs(placement[i] - col) ==
std::abs(i - row)) {
            return false;
        }
    }
    return true;
}
void solveNQueensUtil(std::vector<int>& placement, int row, int&
branchAndBoundSolutions) {
    if (row == N) {
        ++branchAndBoundSolutions;
        return;
    }

    for (int col = 0; col < N; ++col) {
        if (isSafe(placement, row, col)) {
            placement[row] = col;
            solveNQueensUtil(placement, row + 1,
branchAndBoundSolutions);
        }
    }
}

void solveNQueens() {
   std::vector<int> placement(N, 0);
    int branchAndBoundSolutions = 0;

   solveNQueensUtil(placement, 0, branchAndBoundSolutions);

    std::cout << "Метод ветвей и границ: Всего решений найдено - "
<< branchAndBoundSolutions << std::endl;
}
void printSolution(const std::vector<int>& placement) {
```

```
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            std::cout << (placement[i] == j ? 1 : 0) << " ";
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
}
int bruteForceSolutions() {
    int count = 0;
    std::vector<int> placement(N);

    for (int i0 = 0; i0 < N; ++i0) {
        placement[0] = i0;
        for (int i1 = 0; i1 < N; ++i1) {
            placement[1] = i1;
            for (int i2 = 0; i2 < N; ++i2) {
                placement[2] = i2;
                for (int i3 = 0; i3 < N; ++i3) {
                    placement[3] = i3;
                    for (int i4 = 0; i4 < N; ++i4) {
                        placement[4] = i4;
                        for (int i5 = 0; i5 < N; ++i5) {
                            placement[5] = i5;
                            for (int i6 = 0; i6 < N; ++i6) {
                                placement[6] = i6;
                                for (int i7 = 0; i7 < N; ++i7) {
                                    placement[7] = i7;

                                    bool valid = true;
                                    for (int row = 0; row < N;
++row) {
                                        if (!isSafe(placement, row,
placement[row])) {
                                            valid = false;
                                            break;
                                        }
                                    }

                                    if (valid) {
                                        ++count;
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }

    return count;
}

int main() {
    setlocale(0, "Rus");

    // Решение методом ветвей и границ
    auto start = std::chrono::high_resolution_clock::now();
    solveNQueens();
    auto end = std::chrono::high_resolution_clock::now();
    auto duration =
std::chrono::duration_cast<std::chrono::microseconds>(end - start);
```

```
    std::cout << "Время выполнения метода ветвей и границ: " <<
duration.count() << " мкс" << std::endl;

    // Грубая сила
    start = std::chrono::high_resolution_clock::now();
    int bruteForceCount = bruteForceSolutions();
    end = std::chrono::high_resolution_clock::now();
    std::cout << "Грубая сила: Количество решений: " <<
bruteForceCount << std::endl;
    duration =
std::chrono::duration_cast<std::chrono::microseconds>(end - start);
    std::cout << "Время выполнения грубой силы: " <<
duration.count() << " мкс" << std::endl;

    return 0;
}
```

Тестирование: