ilp_grafico.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define TAMANHO_VETOR 100000000

// Função para registrar os dados no arquivo CSV
void registrar_csv(FILE *arquivo, const char *otimizacao, const char *laco, double tempo) {
    if (arquivo) {
        fprintf(arquivo, "%s;%s;%f\n", otimizacao, laco, tempo);
    }
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Uso: %s <Nivel_Otimizacao>\n", argv[0]);
        return 1;
    }
    char *nivel_otimizacao = argv[1];

    FILE *arquivo_csv = fopen("dados_desempenho.csv", "a");
    if (arquivo_csv == NULL) {
        perror("Erro ao abrir o arquivo CSV");
        return 1;
    }

    int *vetor = (int *)malloc(TAMANHO_VETOR * sizeof(int));
    if (vetor == NULL) {
        fprintf(stderr, "Falha na alocacao de memoria\n");
        fclose(arquivo_csv);
        return 1;
    }

    struct timespec inicio, fim;
    double tempo_gasto;

    // --- Laço 1: Inicialização do Vetor ---
    clock_gettime(CLOCK_MONOTONIC, &inicio);
    for (int i = 0; i < TAMANHO_VETOR; i++) {
        vetor[i] = (i % 10) + 1;
    }
    clock_gettime(CLOCK_MONOTONIC, &fim);
    tempo_gasto = (fim.tv_sec - inicio.tv_sec) + (fim.tv_nsec - inicio.tv_nsec) / 1e9;
    printf("Laço 1 (Inicializacao)................: %f segundos\n", tempo_gasto);
    registrar_csv(arquivo_csv, nivel_otimizacao, "Laco_1_Inicializacao", tempo_gasto);

    // --- Laço 2: Soma Acumulativa ---
    long long soma_dependente = 0;
    clock_gettime(CLOCK_MONOTONIC, &inicio);
    for (int i = 0; i < TAMANHO_VETOR; i++) {
```

```
52          soma_dependente += vetor[i];
53      }
54      clock_gettime(CLOCK_MONOTONIC, &fim);
55      tempo_gasto = (fim.tv_sec - inicio.tv_sec) + (fim.tv_nsec - inicio.tv_nsec) / 1e9;
56      printf("Laço 2 (Soma com Dependencia).....: %f segundos\n", tempo_gasto);
57      registrar_csv(arquivo_csv, nivel_otimizacao, "Laco_2_Dependente", tempo_gasto);
58
59      // --- Laço 3: Quebra de Dependências (Fator 4) ---
60      long long s1=0, s2=0, s3=0, s4=0;
61      clock_gettime(CLOCK_MONOTONIC, &inicio);
62      for (int i = 0; i < TAMANHO_VETOR; i += 4) {
63          s1 += vetor[i]; s2 += vetor[i+1]; s3 += vetor[i+2]; s4 += vetor[i+3];
64      }
65      long long soma_total_indep4 = s1 + s2 + s3 + s4;
66      clock_gettime(CLOCK_MONOTONIC, &fim);
67      tempo_gasto = (fim.tv_sec - inicio.tv_sec) + (fim.tv_nsec - inicio.tv_nsec) / 1e9;
68      printf("Laço 3 (Quebra de Dependencia, Fator 4): %f segundos\n", tempo_gasto);
69      registrar_csv(arquivo_csv, nivel_otimizacao, "Laco_3_Fator_4", tempo_gasto);
70
71      // --- Laço 4: Quebra de Dependências (Fator 8) ---
72      long long a1=0, a2=0, a3=0, a4=0, a5=0, a6=0, a7=0, a8=0;
73      clock_gettime(CLOCK_MONOTONIC, &inicio);
74      for (int i = 0; i < TAMANHO_VETOR; i += 8) {
75          a1 += vetor[i];   a2 += vetor[i+1]; a3 += vetor[i+2]; a4 += vetor[i+3];
76          a5 += vetor[i+4]; a6 += vetor[i+5]; a7 += vetor[i+6]; a8 += vetor[i+7];
77      }
78      long long soma_total_indep8 = a1+a2+a3+a4+a5+a6+a7+a8;
79      clock_gettime(CLOCK_MONOTONIC, &fim);
80      tempo_gasto = (fim.tv_sec - inicio.tv_sec) + (fim.tv_nsec - inicio.tv_nsec) / 1e9;
81      printf("Laço 4 (Quebra de Dependencia, Fator 8): %f segundos\n", tempo_gasto);
82      registrar_csv(arquivo_csv, nivel_otimizacao, "Laco_4_Fator_8", tempo_gasto);
83
84
85      // --- NOVA SEÇÃO FINAL ---
86      // Criamos um resultado final para garantir que todas as somas são essenciais.
87      long long total_geral = soma_dependente + soma_total_indep4 + soma_total_indep8;
88      printf("\nVerificacao de Somas (Dependente: %lld, Fator 4: %lld, Fator 8: %lld)\n",
89             soma_dependente, soma_total_indep4, soma_total_indep8);
90
91      free(vetor);
92      fclose(arquivo_csv);
93
94      // O valor de retorno do programa agora depende dos resultados.
95      // O compilador NÃO PODE otimizar isto.
96      return (int)(total_geral % 256);
97  }
```