МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ

«ОПРЕДЕЛЕНИЕ ВРЕМЕНИ РАБОТЫ ПРИКЛАДНЫХ ПРОГРАММ И ИЗУЧЕНИЕ ОПТИМИЗИРУЮЩЕГО КОМПИЛЯТОРА»

студентки 2 курса, группы 21207

Черновской Яны Тихоновны

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

А.Ю. Власенко

Новосибирск 2022

СОДЕРЖАНИЕ

ЦЕЛЬ

1. Изучение методики измерения времени работы подпрограммы.

2. Изучение приемов повышения точности измерения времени работы подпрограммы.

3. Изучение способов измерения времени работы подпрограммы.

4. Измерение времени работы подпрограммы в прикладной программе.

5. Изучение основных функций оптимизирующего компилятора, и некоторых примеров оптимизирующих преобразований и уровней оптимизации.

6. Получение базовых навыков работы с компилятором GCC.

7. Исследование влияния оптимизационных настроек компилятора GCC на время исполнения программы.

ЗАДАНИЕ

1. Написать программу на языке C или C++, содержащую функцию, которая реализует выбранный алгоритм из задания. Программа должна принимать значение N через параметр в командной строке.

2. Проверить правильность работы программы на нескольких тестовых наборах входных данных.

3. Выбрать значение параметра N0 таким, чтобы время работы функции было от 30 до 60 секунд.

4. Программу скомпилировать компилятором g++ с уровнями оптимизации -O0, -O1, -O2, -O3, -Os, -Ofast, -Og под архитектуру процессора x86 (x86-64).

5. Для каждого из семи вариантов компиляции измерить время работы программы при нескольких значениях N (0.5* N0, N0, 1.5* N0).

6. Составить отчет по лабораторной работе. Отчет должен содержать следующее:

a) Титульный лист.

b) Цель лабораторной работы.

c) Вариант задания.

d) Описание методики для определения времени работы программы.

e) Результаты измерения времени работы программы при различных значениях параметра N с уровнями оптимизации -O0, -O1, -O2, -O3, -Os, -Ofast, -Og (лучше в табличном виде).

f) Графики зависимости времени выполнения программы с уровнями оптимизации -O0, -O1, -O2, -O3, -Os, -Ofast, -Og от параметра N.

g) Полный компилируемый листинг реализованной программы, команды для ее компиляции и запуска.

h) Вывод по результатам лабораторной работы

Формулировка задания:

Алгоритм вычисления функции ln(1+x) с помощью разложения в ряд по первым N членам этого ряда:

$$ln(1+x) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots + (-1)^{n+1} \frac{x^n}{n} + \ldots$$

Область сходимости ряда: - 1 ≤ x ≤1 .

ОПИСАНИЕ РАБОТЫ

1. Была написана программа на языке C++, содержащую функцию, которая считает значение натурального логарифма с помощью разложения в ряд по первым N членам этого ряда. Программа принимает значение N и x через параметры в командной строке.

   Сборка проекта:

```
evmpu@comrade:~/21207/Chernovskaya$ g++ -g  lab1/*.cpp -o main.exe -lrt
```

2. Далее были проверены некоторые значения, для проверки корректности работы программы (полученные данные были сравнены с компьютерным вычислением выражения)

```
evmpu@comrade:~/21207/Chernovskaya$ ./main.exe 0.5 50000000
0.3948598226
evmpu@comrade:~/21207/Chernovskaya$ ./main.exe 0.2 500000000
0.1805326933
evmpu@comrade:~/21207/Chernovskaya$ ./main.exe 0.8 500000000
0.5800878356
```

3. Была оценена степень загрузки процессора другими процессами с помощью команды top. Так как степень загрузки процессора не высока, то выбирается произвольный таймер.

```
Tasks: 474 total,   3 running, 470 sleeping,   1 stopped,   0 zombie
%Cpu(s):  8,3 us,  0,0 sy,  0,0 ni, 91,6 id,  0,1 wa,  0,0 hi,  0,0 si,  0,0 st
МиБ Mem :  24064,6 total,  13909,7 free,   1646,7 used,   8508,1 buff/cache
МиБ Swap:   7813,0 total,   7813,0 free,      0,0 used.  22020,8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
2688962 evmpu     20   0   11164   5640   3684 S   0,3   0,0   0:00.28 bash
  42673 root      20   0  815696  24888  17312 S   0,3   0,1  10:11.23 NetworkManager
 665305 evmpu     20   0   12328   4400   3344 R   0,3   0,0   0:03.25 top
 666785 root      20   0       0      0      0 I   0,3   0,0   0:00.23 kworker/0:0-events
3998987 vlasenko  20   0  311724   8064   7140 S   0,3   0,0   0:10.35 gsd-housekeepin
      1 root      20   0  169976  13364   8340 S   0,0   0,1  14:42.36 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.50 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
      7 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0H-events_highpri
     10 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_tasks_rude_
     12 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_tasks_trace
     13 root      20   0       0      0      0 S   0,0   0,0   0:04.43 ksoftirqd/0
     14 root      20   0       0      0      0 I   0,0   0,0   1:31.91 rcu_sched
     15 root      rt   0       0      0      0 S   0,0   0,0   0:03.46 migration/0
     16 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject/0
     18 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
```

4. Для определения времени работы подпрограммы была выбрана библиотечная функция clock_gettime. При условии выполнении программы за промежуток времени 30 - 60 секунд, было подобрано значение N = 7700000000.

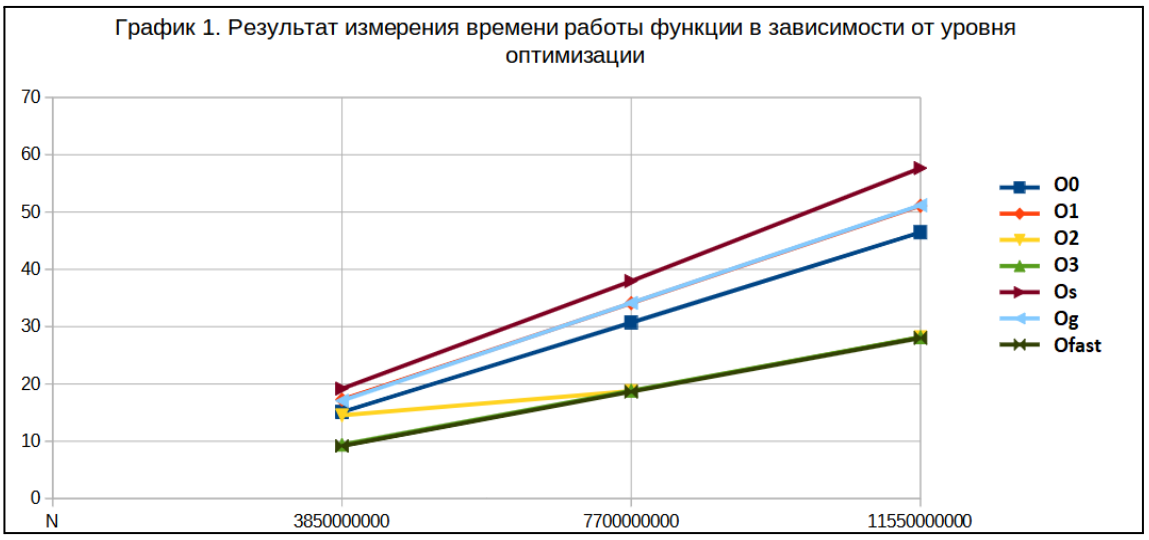
```
evmpu@comrade:~/21207/Chernovskaya$ ./main.exe 7700000000
0.180533
Run №1 x = 0.2 n = 7700000000. Time taken 30.4709
```

5. Далее была скомпилирована программа компилятором g++ с уровнями оптимизации -O0, -O1, -O2, -O3, -Os, -Ofast, -Og

```
evmpu@comrade:~/21207/Chernovskaya$ g++ -g -O0 lab1/*.cpp -o p0.out -lrt
evmpu@comrade:~/21207/Chernovskaya$ g++ -g -O1 lab1/*.cpp -o p1.out -lrt
evmpu@comrade:~/21207/Chernovskaya$ g++ -g -O2 lab1/*.cpp -o p2.out -lrt
evmpu@comrade:~/21207/Chernovskaya$ g++ -g -O3 lab1/*.cpp -o p3.out -lrt
evmpu@comrade:~/21207/Chernovskaya$ g++ -g -Os lab1/*.cpp -o ps.out -lrt
evmpu@comrade:~/21207/Chernovskaya$ g++ -g -Ofast lab1/*.cpp -o pfast.out -lrt
evmpu@comrade:~/21207/Chernovskaya$ g++ -g -Og lab1/*.cpp -o pg.out -lrt
```

6. Для каждого из семи вариантов компиляции было измерено время работы программы при нескольких значениях N (0.5* N0, N0, 1.5* N0)

*Данные, представленные в виде таблице, расположены в приложении 2.



ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были изучены методики измерения времени работы подпрограммы в прикладной программе, приемы повышения ее точности, способы ее измерения, основные функции оптимизирующего компилятора, и некоторые примеры оптимизирующих преобразований и уровней оптимизации, а также влияния оптимизационных настроек компилятора на время исполнения программы.

По первой половине работы можно сделать вывод, что выбор таймера для измерения времени подпрограммы в прикладной программы зависит от степени нагрузки процессора другими процессами (если степень загрузки процессора высока, то выбирается таймер времени процесса, в противном случае выбирается произвольный таймер)

По второй половине работы можно сделать вывод, что в зависимости от выбора уровня оптимизации, могут быть включены оптимизации для уменьшения размера бинарного исполняемого файла и такие оптимизации, уменьшающие время работы программы, которые не сильно замедляют работу компилятора, использование более быстрых и менее точных математических функций и т.д.

ПРИЛОЖЕНИЕ

Приложение 1. Полный листинг программы.

check.h

```
#ifndef CHECK_H
#define CHECK_H

void CheckInput(double x, long long int n);
void CheckArgc(int argc);

#endif
```

check.cpp

```
#include <cstdlib>
#include "check.h"
#include <iostream>
#include <exception>

void CheckInput(double x, long long int n)
{
    if (x <= -1 || x > 1)
    {
        throw std::logic_error("Wrong x value");
    }

    if (n < 1)
    {
        throw std::logic_error("Wrong n value");
    }
}

void CheckArgc(int argc)
{
    if (argc != 3)
    {
        std::cout << "Not enough arguments" << std::endl;
        exit(EXIT_SUCCESS);
    }
}
```

logarithm.cpp

```cpp
#include "check.h"
#include "logarithm.h"
#include <iostream>

double CountLogarithm(double x, long long int n)
{
    CheckInput(x, n);

    double result = 0;

    for (long long int i = 1; i <= n; i++)
    {
        if (i % 2 == 1)
        {
            result += x / i;
        }
        else
        {
            result -= x / i;
        }

        x *= x;
    }

    return result;
}
```

logaritm.h

```cpp
#ifndef LOGARITHM_H
#define LOGARITHM_H

double CountLogarithm(double x, long long int n);

#endif
```

main.cpp

```cpp
#include <iostream>
#include <cstdlib>
#include <exception>
#include <iomanip>
#include <ctime>
#include "check.h"
#include "logarithm.h"

using namespace std;

bool CommandLineWork(int argc, char *argv[]);
void PrintTimeProgramWork(long long int n);

int main(int argc, char *argv[])
{

    if (!CommandLineWork(argc, argv))
    {
        return EXIT_FAILURE;
    }

   // PrintTimeProgramWork(7700000000);

    return EXIT_SUCCESS;
}

void PrintTimeProgramWork(long long int n)
{
    struct timespec start, end;

    double totalTime = 0;       You, 4 days ago • Add first task …
    int runs = 1;

    double x = 0.2;
```

```
35
36      for (int i = 0; i < runs; i++)
37      {
38          clock_gettime(CLOCK_MONOTONIC_RAW, &start);
39          cout << CountLogarithm(0.2, n) << endl;
40          clock_gettime(CLOCK_MONOTONIC_RAW, &end);
41
42          cout << "Run №" << i + 1 << " x = " << x << " n = " << n << ". Time taken ";
43          double time = end.tv_sec - start.tv_sec + 1e-9 * (end.tv_nsec - start.tv_nsec);
44          cout << time << endl;
45          totalTime += time;
46      }
47
48      //cout << "Average time: " << totalTime / runs << " sec" << endl;
49  }
50
51  bool CommandLineWork(int argc, char *argv[])
52  {
53      CheckArgc(argc);
54
55      double x = atof(argv[1]);
56      int n = atoll(argv[2]);
57
58      try
59      {
60          cout << fixed << setprecision(10) << CountLogarithm(x, n) << endl;
61      }
62      catch (const logic_error &e)
63      {
64          cout << e.what() << endl;
65          return false;
```

```
66      }
67      return true;
68  }
```

Приложение 2.

Табл 1. Результаты измерения времени работы функции в зависимости от уровня оптимизации

N	O0	O1	O2	O3	Os	Og	Ofast
3850000000	15,1	17,27	14,51	9,41	19,19	17,08	9,18
7700000000	30,71	34,1	18,75	18,84	37,96	34,15	18,65
11550000000	46,46	51,09	28,14	28,16	57,67	51,19	28,03