МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Программирование рекурсивных алгоритмов

Студент гр. 7383 _____ Александров Р.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2018

Цель работы.

Познакомиться с основными понятиями и приемами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++.

Основные теоретические положения.

Рекурсивным называется объект, содержащий сам себя или определенный с помощью самого себя.

Мощность рекурсии связана с тем, что она позволяет определить бесконечное множество объектов с помощью конечного высказывания. Точно так же бесконечные вычисления можно описать с помощью конечной рекурсивной программы. Рекурсивные алгоритмы лучше всего использовать, когда решаемая задача, вычисляемая функция или обрабатываемая структура данных определены с помощью рекурсии.

Если процедура (функция) P содержит явное обращение к самой себе, она называется прямо рекурсивной. Если P содержит обращение к процедуре (функции) Q, которая содержит (прямо или косвенно) обращение к P, то P называется косвенно рекурсивной.

Многие известные функции могут быть определены рекурсивно. Например, факториал, который присутствует практически во всех учебниках по программированию, а также наибольший общий делитель, числа Фибоначчи, степенная функция и др.

Постановка задачи.

Вариант 1. Для заданных неотрицательных целых n и m вычислить (рекурсивно) биномиальные коэффициенты, пользуясь их определением (см Рисунок 1 – Определение биномиальных коэффициентов).

$$C_n^m = \begin{cases} 1, \text{если } m = 0, n > 0 \text{ или } m = n \geq 0, \\ 0, \text{если } m > n \geq 0, \\ C_{n-1}^{m-1} + C_{n-1}^m \text{ в остальных случаях.} \end{cases}$$

Рисунок 1 – Определение биномиальных коэффициентов

Реализация задачи.

Для решения поставленной задачи в работе были использованы 3 класса: Main, Binom, Helper.

В классе Main определяются функции для считывания данных:

- void consoleRead() - из консоли;
- void fileRead() - из файла.

Пользователю предлагается либо ввести два числа m и n вручную, либо указать текстовый файл, в котором парами находятся нужные числа.

В классе Binom определяются функции рекурсивного вычисления биномиального коэффициента:

- uint64_t count(uint64_t m, uint64_t n) является оберткой над рекурсией и вызывает функцию eval;
- uint64_t eval(uint64_t m, uint64_t n, unsigned long long &half, unsigned long long &full) высчитывает биномиальный коэффициент.

Функция void printUnderscore(unsigned short int i) выводит нижний слеш.

В классе Helper находится ряд вспомогательных функций:

- bool validateFile(std::ifstream &inFile) принимает на вход файловый поток, проверяет соответствующий файл на наличие четного количества цифр и их положительность;
- void outputToFile(std::vector<uint64_t> &resValues) принимает на вход вектор, содержащий результат вычисления биноминального коэффициента для пар чисел, и распечатывает его в файл result.txt;
- void outputToConsole(std::vector<uint64_t> &resValues) принимает на вход вектор, содержащий результат вычисления биноминального коэффициента для пар чисел, и выводит его в консоль.

Сравнение с итеративным решением.

Для проверки целесообразности использования рекурсивной функции была использована версия с циклом for. Результаты показывают, что для вычисления больших значений рекурсия требует намного больше времени.

При значениях m=15, n=30 рекурсивный метод затрачивает 3,22 секунды.

При больших значениях n (m = 15, n = 50) рекурсии требуется более 2 минут.

В обоих случаях версия с циклом счет идет на миллисекунды (конечное и начальное значения clock_t оказывались одинаковыми).

Тестирование программы.

Программа собрана и проверена в операционных системах Ubuntu 18.04 с использованием компилятора g++ и Windows с использованием MinGW. В других ОС и компиляторах тестирование не проводилось. Тесты находятся в приложении А.

Вывод.

В ходе лабораторной работы были получены основные навыки программирования рекурсивных функций на языке C++, изучены понятия и приемы рекурсивного программирования.

Было выявлено, что при больших значениях биноминального коэффициента существует возможность переполнения значения uint64_t (~$2.8*10^{19}$).

По сравнению с итеративным решением рекурсивный метод показывает свою неэффективность, из-за факториального роста сложности затраты по времени оказываются очень существенными.

ПРИЛОЖЕНИЕ А
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Таблица 1 – Тестирование программы

Input	Output
m = 15, n = 30	x = 155117520
m = 7, n = 10	x = 120
m = 6, n = 14	x = 3003
m = 15, n = 4	x = 0
m = -8, n = 6	x = 0
m = -6, n = -14	x = 0
m = 8, n = 8	x = 1
m = 10, n = 16	x = 8008

ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ

main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "main.h"
#include "binom.h"

using namespace std;

void Main::fileRead() {
    string fileName;
    uint64_t value;
    vector<uint64_t> resValues;

    cout << "What`s the file name?" << endl;
    cin >> fileName;
    cout << "--------------" << endl;
    cout << "Reading from " << fileName << endl;
    cout << "--------------" << endl;

    ifstream inFile;
    inFile.open(fileName);
    if (!inFile) {
        cout << "Cannot find this file" << endl;
        cout << endl;
        return;
    }

    /* Check file for work */
    if (!helper.validateFile(inFile)) {
        cout << "Odd number of digits" << endl;
        inFile.close();
        return;
    }

    /* Read file num by num*/
    cout << "-------- Numbers --------" << endl;
    while (inFile >> value) {
        m = value;
        inFile >> value;
        n = value;
        cout << "m = " << m;
        cout << ", n = " << n << endl;
        x = binom.count(m, n);
```

```
        cout << "x = " << x << endl;
        resValues.push_back(x);
    }
    inFile.close();
    cout << "--------------" << endl;

    helper.outputToConsole(resValues);
    helper.outputToFile(resValues);
}

void Main::consoleRead() {
    try {
        cout << "Input m" << endl;
        cin >> m;
        cout << "Input n" << endl;
        cin >> n;
        cout << "-------- Numbers --------" << endl;
        cout << "m = " << m << endl;
        cout << "n = " << n << endl;
        x = binom.count(m, n);
        cout << "x = " << x << endl;
        cout << "--------------" << endl;
    } catch (invalid_argument iae) {
        cout << "Invalid arguments" << endl;
    }
}

void Main::menu() {
    cout << "1. Enter numbers from the txt" << endl;
    cout << "2. Enter numbers from the console" << endl;
    cout << "0. Exit" << endl;
}

int main() {
    cout << "Hello! This program calculates the Binomial coefficient" <<
endl;
    Main main;

    while (true) {
        main.menu();
        cin >> main.choice;
        switch (main.choice) {
            case 1:
                main.fileRead();
                break;
            case 2:
                main.consoleRead();
                break;
            case 0:
                exit(1);
        }
```

```
    }
}
```

main.h

```
#ifndef LAB1_R_H
#define LAB1_R_H
#pragma once

#include <fstream>
#include "helper.h"
#include "binom.h"

class Main {
private:
    Helper helper;
    Binom binom;
    uint64_t m;
    uint64_t n;
    uint64_t x;

public:
    unsigned int choice;

    void consoleRead();

    void fileRead();

    void menu();
};

#endif
```

binom.cpp

```
#include <iostream>
#include "binom.h"

using namespace std;

uint64_t Binom::eval(uint64_t m, uint64_t n, unsigned long long &half,
unsigned long long &full) {
    if (m > n && n >= 0) {
        return 0;
    } else if ((m == 0 && n > 0) || (n >= 0 && m == n)) {
        return 1;
    } else {
        i++;
        printUnderscore(i);
```

```
        cout << "in [" << i << "]" << " with m = " << m << " and n = " <<
n << endl;

        half = eval(m - 1, n - 1, half, full);  // solve a smaller problem
        full = half + eval(m, n - 1, half, full);  // use the solution of
the smaller problem

        printUnderscore(i);
        cout << "out [" << i << "]" << " with m = " << m << " and n = " <<
n << ",  x = " << full << endl;
        i--;

        return full;
    }
}

uint64_t Binom::count(uint64_t m, uint64_t n) {
    unsigned long long h = 0;
    unsigned long long f = 0;
    return eval(m, n, h, f);
}

void Binom::printUnderline(unsigned short int i) {
    for (short j = 1; j <= i; j++) cout << "_";
}
```

binom.h

```
#ifndef LAB1_BINOM_H
#define LAB1_BINOM_H
#pragma once

#include <cstdint>

class Binom {
private:
    unsigned short int i = 0; // iterator
    void printUnderscore(unsigned short int i);
    uint64_t  eval(uint64_t  m,  uint64_t  n,  unsigned  long  long  &half,
unsigned long long &full); // recursive function

public:
    uint64_t count(uint64_t m, uint64_t n); // recursive function
};

#endif
```

helper.cpp

```
#include <fstream>
#include <iostream>
#include "helper.h"

using namespace std;

/* Check for negative numbers and the even sum of digits */
bool Helper::validateFile(ifstream &inFile) {
    long value;
    long count = 0;
    while (inFile >> value) {
        count++;
        if (value < 0) return false;
    }
    cout << count << endl;
    inFile.clear();
    inFile.seekg(0);
    return count % 2 == 0;
}

void Helper::outputToFile(std::vector<uint64_t> &resValues) {
    ofstream outFile;
    outFile.open("result.txt");
    for (uint64_t resValue : resValues) {
        outFile << resValue;
        if (resValue == 0) outFile << " - The answer is 0 because m was
more than n";
        outFile << endl;
    }
    outFile.close();
}

void Helper::outputToConsole(std::vector<uint64_t> &resValues) {
    cout << "Vector consists of " << resValues.size() << " numbers" <<
endl;
    for (uint64_t resValue : resValues) {
        cout << resValue << endl;
    }
}
```

helper.h

```
#ifndef LAB1_HELPER_H
#define LAB1_HELPER_H
#pragma once

#include <fstream>
#include <vector>
```

```
/*
 * The class helps to do interim actions
 */

class Helper {
public:
    bool validateFile(std::ifstream &inFile);

    void outputToFile(std::vector<uint64_t> &resValues);

    void outputToConsole(std::vector<uint64_t> &resValues);
};

#endif
```

Makefile

```
CXX=g++
RM=rm -f
LDFLAGS=-g -Wall

SRCS=main.cpp binom.cpp helper.cpp
OBJS=$(subst .cpp,.o,$(SRCS))

all: main

main: $(OBJS)
	$(CXX) $(LDFLAGS) -o main $(OBJS)

main.o: main.cpp main.h

binom.o: binom.cpp binom.h

helper.o: helper.cpp helper.h

clean:
	$(RM) $(OBJS)

distclean: clean
	$(RM) main
```