Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Отчет по лабораторной работе №5

на тему:

“Индуктивные функции. Обработка файлов ”

по дисциплине “Программирование”

Вариант 5.

Выполнил: Быков И. В.

Группа 6383

Принял: Самойленко В. П.

Санкт-Петербург  
2016 г.

**Цель**

Изучение однопроходных алгоритмов обработки последовательностей на

примере индуктивных функций. Обработка файлов.

**Теоретические сведения.**

Пусть заданы некоторые множества X и Y. Рассмотрим множество функций

F, аргументами которых являются последовательности элементов множества X, а значениями – элементы множества Y. Пусть – пространство всех конечных последовательностей с элементами из множества X длины не менее, чем k. Обозначим через .

Функция называется индуктивной, если можно вычис-лить, зная и , т. е. если существует функция , такая, что для всех и выполняется соотношение .

**Критерий индуктивности.**

Пусть даны две различные последовательности , при этом . Функция f является индуктивной тогда и только тогда, когда для любого x = . Таким образом, для того, чтобы доказать, что функция не является индуктивной, достаточно привести пример и x таких, что и .

Вычислить значение индуктивной функции можно за один проход по последовательности, сохраняя при этом только значение функции на предыдущем шаге.

Для неиндуктивных функций можно построить индуктивное расширение.

То есть найти индуктивную функцию такую, что существует функция

**Обработка текстовых файлов.**

#include<fstream>

fstream – чтение / запись

ifstream (чтение)

ofstream – запись

Пример: откроем на запись файл “printf.txt”, запишем туда информацию и закроем:

ofstream outfile; // назвали объект

outfile.open(“printf.txt”);//связали объект с файлом

outfile << “Hello!”; // записали строку в файл

outfile.close() // закрыли файл

Режимы открытия файлов:

|  |  |
| --- | --- |
| Константа | Значение |
| **ios\_base::in** | открыть для чтения |
| **ios\_base::out** | открыть для записи |
| **ios\_base::ate** | при открытии переместить указатель в конец |
| **ios\_base::app** | открыть для записи в конец |
| **ios\_base::trunk** | удалить содержимое |
| **ios\_base::binary** | открытие в двоичном режиме |

Реализуется так: outfile.open(“printf.txt” , ios\_base::\*)

**Задание**

Найти размах (разность максимального и минимального) значений элементов последовательности:

**Требования к лабораторной работе.**

1) Алгоритм решения задачи должен быть однопроходным и не должен использовать дополнительную память, пропорциональную размеру входного файла (использование массивов для хранения входной последовательности запрещено);

2) Рекомендуется использовать схему вычисления индуктивной функции, в

которой на каждом шаге основного цикла из входного файла читается и обрабатывается один элемент последовательности;

3) Программа должна выводить поочередно обрабатываемые элементы

входного файла, а также другие (промежуточные) данные в виде, позволяющем

проанализировать и понять процесс формирования окончательного результата;

4) Выходные данные должны выводиться на экран и в выходной файл (в режиме формирования отчета);

5) Требуется протестировать основные особые случаи входных данных

(например, входной файл пуст или содержит лишь один элемент), корректную обработку последнего элемента файла, проверить, учитывается ли наличие стационарного значения индуктивной функции и т. п.

**Проверка на индуктивность.**

Очевидно, что функция не является индуктивной. Докажем это по критерию индуктивности. Возьмем и для которых выполняется условие . Пусть x = 8, тогда

.. Следовательно, функция не является индуктивной.

Построим индуктивное расширение – , где - минимальное и максимальное среди текущих чисел. При этом:

**Таблица экспериментальных значений.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Последовательность в исходном файле | Промежуточный и итоговый результат вида (i , x, amp\_rec , cur\_max , cur\_min ) , где i – порядковый номер элемента, amp\_rec - текущий максимальный размах, cur\_max – текущий максимум, cur\_min – текущий минимум | | | | |
| 1 | «файл пуст» | Входной файл пуст! | | | | |
| 2 | 12 | Введен только один член последовательности! | | | | |
| 3 |  | i | x | f | cur\_max | cur\_min |
| 3 | 1 | 3 | 0 | 3 | 3 |
| 2 | 2 | 2 | 1 | 3 | 2 |
| 0 | 3 | 0 | 3 | 3 | 0 |
| -1 | 4 | -1 | 4 | 3 | -1 |
| 2 | 5 | 2 | 4 | 3 | -1 |
|  | Число элементов =5  Максимальный размах =4 | | | | |

**Спецификации функций.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Описание | Предусловие | Постусловие | Что возвращает? |
| void Find\_Amplitude(int x, short& i, int& amp\_rec, ifstream& infile , ofstream& outfile); | Функция нахождения максимального размаха amp\_rec = abs(cur\_max - cur\_min) |  | i > 1 - последний элемент , amp\_rec >=0 - максимальный размах | Ничего |
| void Output\_File(short i , int amp\_rec , ofstream& outfile); | Функция вывода итогового результата в outfile | i > 1 - номер последнего элемента, amp\_rec >=0 - максимальный размах | Нет | Ничего |
| void Output\_Screen(short i , int amp\_rec); | Функция вывода итогового результата на экран | i > 1 - номер последнего элемента, amp\_rec >=0 - максимальный размах | Нет | Ничего |

**Описание алгоритма.**

void Find\_Amplitude(int x, short& i, int& amp\_rec, ifstream& infile , ofstream& outfile)

{

int cur\_min = INT\_MAX; //задаем минимуму максимальное значение

int cur\_max = INT\_MIN; //задаем максимуму минимальное значение

while (infile >> x) // cчитываем x, пока можем считывать

{

i++; // номер текущего элемента

cur\_max = max(cur\_max , x); // находим текущий максимум.

// Находится, выбирая между предыдущим максимумом и текущим

// элементом последовательности.

cur\_min = min(cur\_min , x); // текущий минимум.

// Находится аналогично текущему максимуму

amp\_rec = abs(cur\_max - cur\_min); // текущий наибольший размах.

// Это модуль разности между минимумом и максимумом

}

return;}

**Текст программы с комментариями.**

#include <iostream> // cin , cout etc

#include <fstream> // ifstream , outstream etc

#include <iomanip> // setw(a)

#include <algorithm> // abs(x,y)

#include <climits> // INT\_MAX, INT\_MIN

using namespace std;

/\* Функция нахождения максимального размаха amp\_rec = abs(cur\_max - cur\_min)

Считаем текущий MAX и MIN из infile, находим текущий максимальный размах и выводим промежуточный результат на экран и в outfile

Предусловие x - целое

Постусловие i > 1 - последний элемент , amp\_rec >=0 - максимальный размах \*/

void Find\_Amplitude(int x, short& i, int& amp\_rec, ifstream& infile , ofstream& outfile);

/\* Функция вывода итогового результата в outfile

Предусловие i > 1 - номер последнего элемента, amp\_rec >=0 - максимальный размах\*/

void Output\_File(short i , int amp\_rec , ofstream& outfile);

/\* Функция вывода итогового результата на экран

Предусловие i > 1 - номер последнего элемента , amp\_rec >=0 - максимальный размах \*/

void Output\_Screen(short i , int amp\_rec);

int main()

{

setlocale(LC\_CTYPE , "rus");

ifstream infile("fscanf.txt");

ofstream outfile("fprintf.txt");

if (!infile.is\_open())

{

cout << "Входный файл не открыт!" << endl;

outfile << "Входной файл не открыт" << endl;

return 0;

}

if (!outfile.is\_open())

{

cout << "Файл вывода не открыт!" << endl;

return 0;

}

int x;

int amp\_rec = 0;

short i = 0;

Find\_Amplitude(x , i , amp\_rec , infile , outfile);

Output\_File(i , amp\_rec ,outfile);

Output\_Screen(i , amp\_rec);

infile.close();

outfile.close();

return 0;

}

void Find\_Amplitude(int x, short& i, int& amp\_rec, ifstream& infile , ofstream& outfile)

{

int cur\_min = INT\_MAX;

int cur\_max = INT\_MIN;

while (infile >> x)

{

i++;

cur\_max = max(cur\_max , x); // текущий максимум

cur\_min = min(cur\_min , x); // текущий минимум

amp\_rec = abs(cur\_max - cur\_min); // текущий наибольший размах

cout << setw(2) << i << ". x = " << setw(4) << x << "; amp\_rec = " << setw(4) << amp\_rec << ", текущий MAX = " << setw(4) << cur\_max << ", текущий MIN =" << setw(4) << cur\_min << endl;

outfile << setw(2) << i << ". x = " << setw(4) << x << "; amp\_rec = " << setw(4) << amp\_rec << ", текущий MAX = " << setw(4) << cur\_max << ", текущий MIN =" << setw(4) << cur\_min << endl;

}

return;

}

void Output\_Screen(short i , int amp\_rec)

{

if (i==0)

cout << "Входной файл пуст!" << endl;

else if (i==1)

cout << "Введен только один член последовательности!" << endl;

else

cout << endl << "Число элементов в последовательности = " << i << endl << "Максимальный размах: " << amp\_rec << endl;

return;

}

void Output\_File(short i , int amp\_rec , ofstream& outfile)

{

if (i==0)

outfile << "Входной файл пуст!" << endl;

else if (i==1)

outfile << "Введен только один член последовательности!" << endl;

else

outfile << endl << "Число элементов в последовательности = " << i << endl << "Максимальный размах: " << amp\_rec << endl;

return;

}

**Результат работы программы.**

(Вывод в файл «fprintf.txt» и вывод на экран одинаковы)

Ввод: «пустой файл»

Вывод: Входной файл пуст!

Ввод: 12

Вывод: Введен только один член последовательности!

Ввод: 3 2 0 -1 2

Вывод:

1. x = 3; amp\_rec = 0, текущий MAX = 3, текущий MIN = 3

2. x = 2; amp\_rec = 1, текущий MAX = 3, текущий MIN = 2

3. x = 0; amp\_rec = 3, текущий MAX = 3, текущий MIN = 0

4. x = -1; amp\_rec = 4, текущий MAX = 3, текущий MIN = -1

5. x = 2; amp\_rec = 4, текущий MAX = 3, текущий MIN = -1

Число элементов в последовательности = 5

Максимальный размах: 4

**Вывод.**

Я изучил однопроходные алгоритмы обработки последовательностей на примере индуктивных функций. Мною был разработан и реализован однопроходный алгоритм нахождения максимального размаха заданной последовательности. Так же ознакомился с обработкой файлов. Научился открывать/закрывать файлы, читать и записывать информацию.