Для подсчета количества возможных номеров используется обход по всем возможным номерам, пропуская номера, не удовлетворяющие условиям из файла:

//обходим все возможные номера

for(PhoneNumber[0]=0;PhoneNumber[0]<10;PhoneNumber[0]++)

{

//на каждом уровне проверяем, удовлетворяет ли данная комбинация цифр условиям

//если нет, то идем дальше

if(!analyze(Func[0],0)) continue;

for(PhoneNumber[1]=0;PhoneNumber[1]<10;PhoneNumber[1]++)

{

if(!analyze(Func[1],1)) continue;

for(PhoneNumber[2]=0;PhoneNumber[2]<10;PhoneNumber[2]++)

{

if(!analyze(Func[2],2)) continue;

for(PhoneNumber[3]=0;PhoneNumber[3]<10;PhoneNumber[3]++)

{

if(!analyze(Func[3],3)) continue;

for(PhoneNumber[4]=0;PhoneNumber[4]<10;PhoneNumber[4]++)

{

if(!analyze(Func[4],4)) continue;

for(PhoneNumber[5]=0;PhoneNumber[5]<10;PhoneNumber[5]++)

{

if(!analyze(Func[5],5)) continue;

//если дошли сюда - значит //комбинация удовлетворительная, прибавляем единицу

Num\_count++;

}

}

}

}

}

}

Функция анализа удовлетворительности текущей комбинации выглядит след. образом:

bool analyze(const vector<condition>& vec,int index)

{

if(!vec.empty()){

My\_iterator it = vec.begin();

while(it!=vec.end())

{

//проходим по массиву условий для тек. позиции

if(it->first(&PhoneNumber[index],it->second)) it++;

//если условие не выполняется - возвращаем 0

else return 0;

}

}

//если пришли сюда - условия выполняются

return 1;

}

Происходит последовательный обход вектора условий для текущего индекса телефонного номера, который заполняется при разборе данных во время чтения из файла:

//чтение из файла

bool rfile(const string& name, vector<condition>\* Vec)

{

ifstream file;

file.open(name);

int a,b,i;

char c[2];

if(file.is\_open()) {

while(!file.eof())

{

i = -1;

file >> a;//

if(file.fail()) break;

do{

file.get(c[++i]);

}

while(!isdigit(c[i]));

file.putback(c[i]);

file >> b;

cout << a << " " << c[0] << " " << c[1] << " Sec chis" << b << endl;

//в зависимости от того, какой попался символ

//<,=,> делаем выбор функций-предикатов

switch (c[0]) {

case '<':

if(c[1] == '>')

{

(a < b) ? Vec[b-1].push\_back(condition(notequal,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(notequal,&PhoneNumber[b-1]));

}

else if(c[1] == '=')

{

(a < b) ? Vec[b-1].push\_back(condition(numhighequal,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(numlessequal,&PhoneNumber[b-1]));

}

else {

(a < b) ? Vec[b-1].push\_back(condition(numhigh,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(numless,&PhoneNumber[b-1]));

}

break;

case '=':

(a < b) ? Vec[b-1].push\_back(condition(equal,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(equal,&PhoneNumber[b-1]));

break;

case '>':

if(c[1] == '=')

{

(a < b) ? Vec[b-1].push\_back(condition(numlessequal,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(numhighequal,&PhoneNumber[b-1]));

}

else {

(a < b) ? Vec[b-1].push\_back(condition(numless,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(numhigh,&PhoneNumber[b-1]));

}

break;

default:

break;

}

//if(file.eof()) break;

}

file.close ();

return 1;

}

else

{

cerr << "File not found" << endl;

return 0;

}

}

Исходный код программы с комментариями:

#include <iostream>

#include <ctime>

#include <vector>

#include <fstream>

#include <windows.h>

using namespace std;

const int size = 6;

//структура для хранения номера

int PhoneNumber[size];

int Num\_count = 0;

//массивы предикатов

typedef bool (\*predicat)(int \*it1, int \*it2);

//синонимы типам для удобства

typedef pair<predicat,int\*> condition;

typedef vector<condition>::const\_iterator My\_iterator;

//функция '='

bool equal(int \*it1, int \*it2)

{

return \*it1==\*it2;

}

//функция '<'

bool numless(int \*it1, int \*it2)

{

return (\*it1)<(\*it2);

}

//функция '>'

bool numhigh(int \*it1, int \*it2)

{

return (\*it1)>(\*it2);

}

//функция '>='

bool numhighequal(int \*it1, int \*it2)

{

return !numless(it1,it2);

}

//функция '<='

bool numlessequal(int \*it1, int \*it2)

{

return !numhigh(it1,it2);

}

//функция '!='

bool notequal(int \*it1, int \*it2)

{

return !equal(it1,it2);

}

//чтение из файла

bool rfile(const string& name, vector<condition>\* Vec)

{

ifstream file;

file.open(name);

int a,b,i;

char c[2];

if(file.is\_open()) {

while(!file.eof())

{

i = -1;

file >> a;//

if(file.fail()) break;

do{

file.get(c[++i]);

}

while(!isdigit(c[i]));

file.putback(c[i]);

file >> b;

cout << a << " " << c[0] << " " << c[1] << " Sec chis" << b << endl;

//в зависимости от того, какой попался символ

//<,=,> делаем выбор функций-предикатов

switch (c[0]) {

case '<':

if(c[1] == '>')

{

(a < b) ? Vec[b-1].push\_back(condition(notequal,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(notequal,&PhoneNumber[b-1]));

}

else if(c[1] == '=')

{

(a < b) ? Vec[b-1].push\_back(condition(numhighequal,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(numlessequal,&PhoneNumber[b-1]));

}

else {

(a < b) ? Vec[b-1].push\_back(condition(numhigh,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(numless,&PhoneNumber[b-1]));

}

break;

case '=':

(a < b) ? Vec[b-1].push\_back(condition(equal,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(equal,&PhoneNumber[b-1]));

break;

case '>':

if(c[1] == '=')

{

(a < b) ? Vec[b-1].push\_back(condition(numlessequal,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(numhighequal,&PhoneNumber[b-1]));

}

else {

(a < b) ? Vec[b-1].push\_back(condition(numless,&PhoneNumber[a-1])):

Vec[a-1].push\_back(condition(numhigh,&PhoneNumber[b-1]));

}

break;

default:

break;

}

//if(file.eof()) break;

}

file.close ();

return 1;

}

else

{

cerr << "File not found" << endl;

return 0;

}

}

bool analyze(const vector<condition>& vec,int index)

{

if(!vec.empty()){

My\_iterator it = vec.begin();

while(it!=vec.end())

{

//проходим по массиву условий для тек. позиции

if(it->first(&PhoneNumber[index],it->second)) it++;

//если условие не выполняется - возвращаем 0

else return 0;

}

}

//если пришли сюда - условия выполняются

return 1;

}

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

{

vector<condition> \*Func = new vector<condition>[size];

if(!rfile("input.txt",Func)) return 0;

//время выполнения

int start = clock();

//обходим все возможные номера

for(PhoneNumber[0]=0;PhoneNumber[0]<10;PhoneNumber[0]++)

{

//на каждом уровне проверяем, удовлетворяет ли данная комбинация цифр условиям

//если нет, то идем дальше

if(!analyze(Func[0],0)) continue;

for(PhoneNumber[1]=0;PhoneNumber[1]<10;PhoneNumber[1]++)

{

if(!analyze(Func[1],1)) continue;

for(PhoneNumber[2]=0;PhoneNumber[2]<10;PhoneNumber[2]++)

{

if(!analyze(Func[2],2)) continue;

for(PhoneNumber[3]=0;PhoneNumber[3]<10;PhoneNumber[3]++)

{

if(!analyze(Func[3],3)) continue;

for(PhoneNumber[4]=0;PhoneNumber[4]<10;PhoneNumber[4]++)

{

if(!analyze(Func[4],4)) continue;

for(PhoneNumber[5]=0;PhoneNumber[5]<10;PhoneNumber[5]++)

{

if(!analyze(Func[5],5)) continue;

//если дошли сюда - значит комбинация удовлетворительная, прибавляем единицу

Num\_count++;

}

}

}

}

}

}

//останов

int stop = clock();

cout << ((float)(stop-start)) / CLOCKS\_PER\_SEC << " seconds" << endl;

cout << "Count is " << Num\_count;

return 0;

}