

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

**Лабораторная работa 4**

Выполнил:

Гр. ВКБ 31

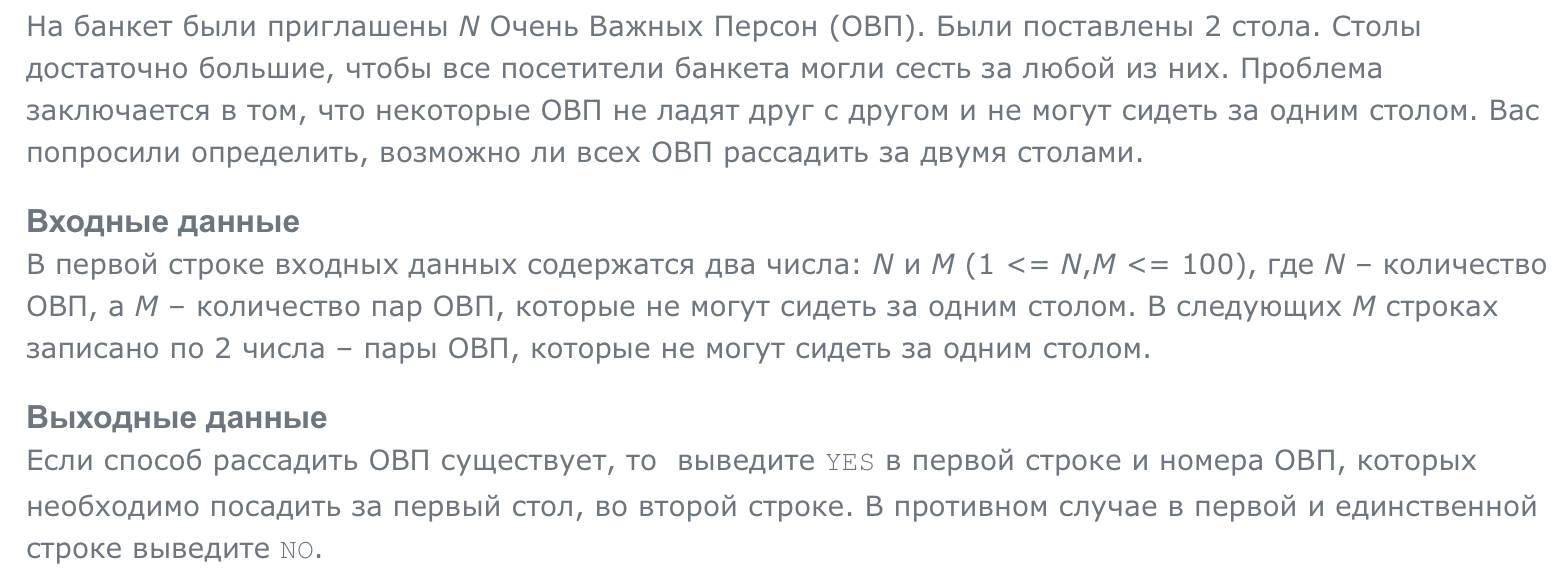
Целиков А.О.

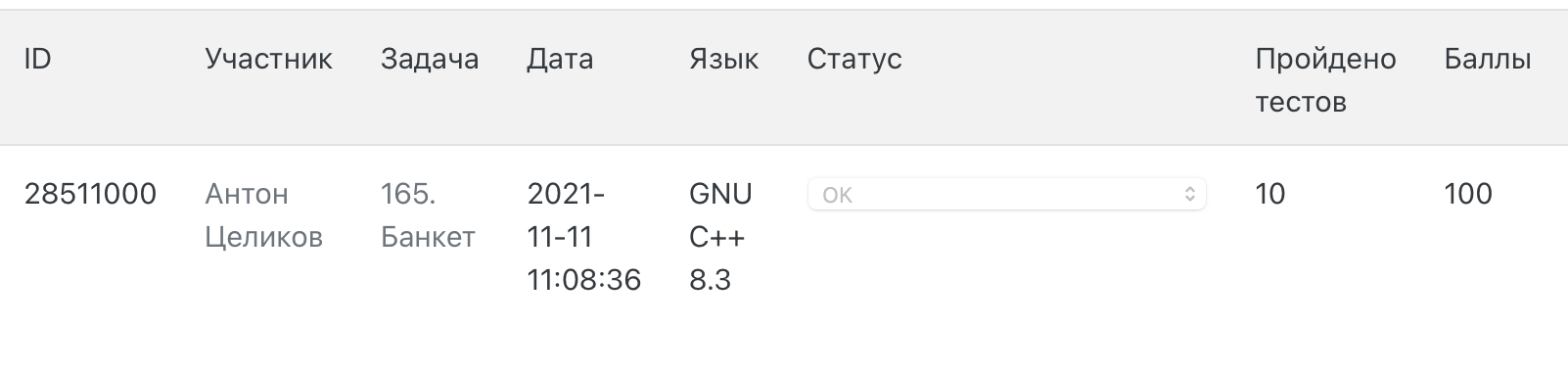
Проверил:

Савельев В.А.

Ростов-на-Дону,

2021г.





Листинг:

#include <algorithm>

#include <fstream>

#include <iostream>

#include <list>

#include <map>

#include <set>

#include <string>

#include <vector>

typedef std::string T\_str;

typedef size\_t T\_person;

typedef std::list<T\_person> T\_bad\_persons;

typedef std::map<T\_person, T\_bad\_persons> T\_bad\_persons\_for\_person;

const T\_person PERSON\_MIN = 1;

enum T\_table

{

NO\_TABLE = 0,

TABLE\_1,

TABLE\_2

};

typedef std::map<T\_person, T\_table> T\_table\_of\_person;

bool successfully\_depth\_first\_search

(

T\_person person,

const T\_bad\_persons\_for\_person& bad\_persons\_for\_person,

T\_table\_of\_person& table\_of\_person

)

{

if(table\_of\_person[person] == NO\_TABLE)

{

table\_of\_person[person] = TABLE\_1;

}

T\_bad\_persons\_for\_person::const\_iterator

person\_and\_bad\_persons\_it = bad\_persons\_for\_person.find(person);

if( person\_and\_bad\_persons\_it == bad\_persons\_for\_person.end() )

{

return true;

}

T\_bad\_persons bad\_persons = person\_and\_bad\_persons\_it->second;

for(

T\_bad\_persons::const\_iterator

bad\_person\_it = bad\_persons.begin();

bad\_person\_it != bad\_persons.end();

++bad\_person\_it

)

{

if(

table\_of\_person[\*bad\_person\_it] == table\_of\_person[person]

)

{

return false;

}

else if(

table\_of\_person[\*bad\_person\_it] == NO\_TABLE

)

{

table\_of\_person[\*bad\_person\_it] = table\_of\_person[person] == TABLE\_1

? TABLE\_2

: TABLE\_1;

if(

!successfully\_depth\_first\_search

(

\*bad\_person\_it,

bad\_persons\_for\_person,

table\_of\_person

)

)

{

return false;

}

}//else if

}//for

return true;

}

bool successfully\_to\_seat

(

const T\_bad\_persons\_for\_person& bad\_persons\_for\_person,

T\_table\_of\_person& table\_of\_person

)

{

for(T\_table\_of\_person::iterator person\_and\_table\_it = table\_of\_person.begin();

person\_and\_table\_it != table\_of\_person.end(); ++person\_and\_table\_it)

{

if(person\_and\_table\_it->second != NO\_TABLE) continue;

if(

!successfully\_depth\_first\_search

(

person\_and\_table\_it->first,

bad\_persons\_for\_person,

table\_of\_person

)

)

{

return false;

}

}

return true;

}

void input\_persons\_data

(

const T\_str& ifile\_name,

T\_table\_of\_person& table\_of\_person,

T\_bad\_persons\_for\_person& bad\_persons\_for\_person

)

{

std::ifstream ifile( ifile\_name.c\_str() );

int persons\_total = 0;

ifile >> persons\_total;

for(T\_person person = PERSON\_MIN; person <= persons\_total; ++person)

{

table\_of\_person[person] = NO\_TABLE;

}

int persons\_pairs\_total = 0;

ifile >> persons\_pairs\_total;

for(int i = 0; i < persons\_pairs\_total; ++i)

{

int L = 0;

ifile >> L;

int R = 0;

ifile >> R;

bad\_persons\_for\_person[L].push\_back(R);

bad\_persons\_for\_person[R].push\_back(L);

}

}

void print\_persons\_placement\_at\_tables

(

const T\_str& ofile\_name,

bool persons\_are\_successfully\_seated,

const T\_table\_of\_person& table\_of\_person

)

{

std::ofstream ofile( ofile\_name.c\_str() );

ofile << (

persons\_are\_successfully\_seated

? "YES"

: "NO"

)

<< std::endl;

if(persons\_are\_successfully\_seated)

{

for(T\_table\_of\_person::const\_iterator person\_and\_table\_it = table\_of\_person.begin();

person\_and\_table\_it != table\_of\_person.end(); ++person\_and\_table\_it)

{

if(person\_and\_table\_it->second == TABLE\_1)

{

ofile << person\_and\_table\_it->first

<< " ";

}

}

ofile << std::endl;

}

}

int main()

{

const T\_str ifile\_name = "input.txt";

const T\_str ofile\_name = "output.txt";

T\_bad\_persons\_for\_person bad\_persons\_for\_person;

T\_table\_of\_person table\_of\_person;

input\_persons\_data

(

ifile\_name,

table\_of\_person,

bad\_persons\_for\_person

);

bool bool\_res = successfully\_to\_seat

(

bad\_persons\_for\_person,

table\_of\_person

);

print\_persons\_placement\_at\_tables

(

ofile\_name,

bool\_res,

table\_of\_person

);

}