Федеральное государственное бюджетное образовательное учреждение высшего образования "Белгородский государственный технологический университет им. В.Г. Шухова"

Кафедра программного обеспечения вычислительной техники и автоматизированных систем.

Расчётно-графическое задание

Обфускация строк кода в C++.

Выполнил:
Студент группы КБ-211

Коренев Д.Н.

Принял:

Осипов О.В.

Белгород 2023

Цель работы: изучение способов обфускации кода, написать обфускатор строк кода на C++.

Выполнение

Обфускатор кода - это программа или инструмент, который превращает исходный код программы в более сложный и непонятный вид, чтобы защитить его от копирования, изменения или взлома. Обфускация кода может быть выполнена на разных уровнях: на уровне исходных текстов, на уровне машинного кода или на уровне промежуточного кода. Обфускация кода может иметь разные цели, такие как улучшение производительности, оптимизация размера файла, демонстрация возможностей языка или квалификация программиста.

Код программы:

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <map>
#include <vector>
#include <random>
#include <regex>
using namespace std;

const string lexems[] = {
    "auto", "break", "case", "char", "const", "continue", "default", "do", "double",
    "else", "enum", "extern", "float", "for", "goto", "if", "int", "long",
"register",
    "return", "short", "signed", "sizeof", "static", "struct", "switch", "typedef",
    "union", "unsigned", "void", "volatile", "while", "asm", "bool", "catch",
"class",
    "const_cast", "delete", "dynamic_cast", "explicit", "export", "false", "friend",
    "inline", "mutable", "namespace", "new", "operator", "private", "protected",
    "public", "reinterpret_cast", "static_cast", "template", "this", "throw",
"true",
    "try", "typeid", "typename", "using", "virtual", "wchar_t", "alignas",
"alignof",
    "char16_t", "char32_t", "constexpr", "decltype", "noexcept", "nullptr",
"static_assert",
    "thread_local", "override", "final", "import", "module", "transaction_safe",
"transaction_safe_dynamic",
    "atomic_cancel", "atomic_commit", "atomic_noexcept", "synchronized", "export",
"module", "import",
    "concept", "requires", "co_await", "co_return", "co_yield", "reflexpr",
"sizeof", "alignof", "typeid",
    "decltype", "static_assert", "noexcept", "template", "typename", "using",
"export", "module", "import",
```

```
    "concept", "requires", "co_await", "co_return", "co_yield", "reflexpr",
"sizeof", "alignof", "typeid",
    "decltype", "static_assert", "noexcept", "template", "typename", "using",
"export", "module", "import",
    "concept", "requires", "co_await", "co_return", "co_yield", "reflexpr",
"sizeof", "alignof", "typeid",
    "decltype", "static_assert", "noexcept", "template", "typename", "using",
"export", "module", "import",
    "concept", "requires", "co_await", "co_return", "co_yield", "reflexpr",
"sizeof", "alignof", "typeid", "iterator", "initializer_list", "other", "list",
"size_t", "vector", "map", "set", "string", "regex", "random_device", "mt19937",
"uniform_int_distribution", "sregex_iterator", "smatch", "endl", "cin", "std",
"cout", "ifstream", "ofstream", "min", "max", "abs", "main"};

// Функция генерации случайного имени переменной заданной длины
string generateRandomName(int length)
{
    static const char alphanum[] = "O0oGgqQ";
    static const char start[] =
        "OooGgqQ";

    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(0, sizeof(alphanum) - 2);

    stringstream ss;
    ss << start[dis(gen)];
    for (int i = 0; i < length; ++i)
    {
        ss << alphanum[dis(gen)];
    }
    return ss.str();
}

// Функция удаления комментариев из текста
string removeComments(const string &code)
{
    regex commentRegex("/\\*([^*]|(\\*+[^*/]))*\\*+/|//[^\\n]*");
    return regex_replace(code, commentRegex, "");
}

// Функция обфускации кода
string obfuscateCode(const string &code)
{
    map<string, string> variableMap;
    string result = removeComments(code);

    // Убираем все табуляции, переносы строк и заменяем множественные пробелы на
одиночные
    result.erase(remove(result.begin(), result.end(), '\t'), result.end());
```

```
    result.erase(remove(result.begin(), result.end(), '\n'), result.end());
    regex spaceRegex("\\s+");
    result = regex_replace(result, spaceRegex, " ");

    // Регулярное выражение для поиска текста в кавычках
    regex stringRegex("\"(?:\\\\.|[^\"])*\"");

    // Находим все строки в кавычках и временно заменяем их временно
    string placeholder = "##STRING##";
    vector<string> stringsToPreserve;
    sregex_iterator stringIter(result.begin(), result.end(), stringRegex);
    sregex_iterator stringEnd;
    while (stringIter != stringEnd)
    {
        smatch match = *stringIter;
        stringsToPreserve.push_back(match.str());
        result.replace(match.position(), match.length(), placeholder);
        ++stringIter;
    }

    // Проводим обфускацию переменных
    regex
variableRegex("(\\b(?:int|float|double|bool|char|string|auto)\\s+)(\\b(?!main\\b)\\w
+)");
    sregex_iterator iter(result.begin(), result.end(), variableRegex);
    sregex_iterator end;

    while (iter != end)
    {
        smatch match = *iter;
        if (match.size() == 3)
        {
            string dataType = match[1];
            string varName = match[2];

            // Проверяем, что переменная не встречается внутри строк
            bool isInString = false;
            for (const auto &str : stringsToPreserve)
            {
                if (str.find(varName) != string::npos)
                {
                    isInString = true;
                    break;
                }
            }

            if (!isInString)
            {
                string newName = generateRandomName(32);
                variableMap[varName] = newName;
```

```
                regex replaceRegex("\\b" + dataType + varName + "\\b");
                result = regex_replace(result, replaceRegex, dataType + newName);
            }
        }
        ++iter;
    }

    // Восстанавливаем строки обратно в текст
    for (size_t i = 0; i < stringsToPreserve.size(); ++i)
    {
        size_t pos = result.find(placeholder);
        result.replace(pos, placeholder.length(), stringsToPreserve[i]);
    }

    for (const auto &pair : variableMap)
    {
        regex variableUsageRegex("\\b" + pair.first + "\\b");
        result = regex_replace(result, variableUsageRegex, pair.second);
    }

    return result;
}

int main()
{
    // Открываем файл .txt для чтения
    ifstream inputFile("input.txt");
    if (!inputFile.is_open())
    {
        cout << "Cannot open input file!" << endl;
        return 1;
    }

    // Считываем содержимое файла в строку
    stringstream buffer;
    buffer << inputFile.rdbuf();
    string originalContent = buffer.str();
    inputFile.close();

    string toObfuscate = originalContent;
    // Убираем из toObfuscate все include define строки и using и помещаем их в
отдельную строку
    string essentials;

    // Убираем все include строки <> ""
    regex includeRegex("#include\\s+[\"<][^\">]+[\">]");
    toObfuscate = regex_replace(toObfuscate, includeRegex, "");

    // Убираем все define строки
```

```
    regex defineRegex("#define\\s+\\w+(\\(.*\\))?\\s+.*");
    toObfuscate = regex_replace(toObfuscate, defineRegex, "");

    // Убираем все using строки
    regex
usingRegex("using\\s+\\w+(::\\w+)*(<[^>]*>)?\\s*=\\s*\\w+(::\\w+)*(<[^>]*>)?;");
    toObfuscate = regex_replace(toObfuscate, usingRegex, "");

    // Находим все include строки и помещаем их в essentials
    sregex_iterator includeIter(originalContent.begin(), originalContent.end(),
includeRegex);
    sregex_iterator includeEnd;

    while (includeIter != includeEnd)
    {
        smatch match = *includeIter;
        essentials += match.str() + "\n";
        ++includeIter;
    }

    // Находим все define строки и помещаем их в essentials
    sregex_iterator defineIter(originalContent.begin(), originalContent.end(),
defineRegex);
    sregex_iterator defineEnd;

    while (defineIter != defineEnd)
    {
        smatch match = *defineIter;
        essentials += match.str() + "\n";
        ++defineIter;
    }

    // Находим все using строки и помещаем их в essentials
    sregex_iterator usingIter(originalContent.begin(), originalContent.end(),
usingRegex);
    sregex_iterator usingEnd;

    while (usingIter != usingEnd)
    {
        smatch match = *usingIter;
        essentials += match.str() + "\n";
        ++usingIter;
    }

    auto lexemDict = map<string, string>();
    for (const auto &lexem : lexems)
    {
        lexemDict[lexem] = generateRandomName(32);
    }
```

```
    string llex;
    for (const auto &pair : lexemDict)
    {
        llex += "#define " + pair.second + " " + pair.first + "\n";
    }

    // Обфусцируем содержимое файла
    string obfuscated = obfuscateCode(toObfuscate);

    for (const auto &pair : lexemDict)
    {
        regex lexemRegex("\\b" + pair.first + "\\b");
        obfuscated = regex_replace(obfuscated, lexemRegex, pair.second);
    }

    obfuscated = obfuscated.substr(3);

    string obfuscatedContent = essentials + llex + obfuscated;

    // Открываем файл для записи обфусцированного содержимого
    ofstream outputFile("output.cpp");
    if (!outputFile.is_open())
    {
        cout << "Cannot open output file!" << endl;
        return 1;
    }

    // Записываем обфусцированное содержимое в файл
    outputFile << obfuscatedContent;
    outputFile.close();

    cout << "Obfuscation complete!" << endl;
    return 0;
}
```

Пример работы №1 (swap-функция):

Оригинальный код:

```
#include <iostream>
using namespace std;

int main()
{
    int a = 5, b = 10, temp;

    cout << "Before swapping." << endl;
    cout << "a = " << a << ", b = " << b << endl;

    temp = a;
    a = b;
```

```
    b = temp;

    cout << "\nAfter swapping." << endl;
    cout << "a = " << a << ", b = " << b << endl;

    return 0;
}
```

```
Before swapping.
a = 5, b = 10

After swapping.
a = 10, b = 5


...Program finished with exit code 0
Press ENTER to exit console.
```

Измененный код:

```
#include <iostream>
#define oqqoqqO00QGoqqG0Oq0qOq0QgOGooQGgQ abs
#define goo0gOoqo0GGGqoGQGg0OgooOO0Gg0OGG alignas
#define og0oqoqOooqOQQO000ggOQgo0QGgoQqoQ alignof
#define gGQogq0gGqQgog0qgQgQOOoG0gQ0GGqog asm
#define qGqGO0oqOG0qqqOGGoQoGgGOGogogQQo0 atomic_cancel
#define QqOG0oGO00oO0OGgoGq0GqggoqOqooOoG atomic_commit
#define OoOgOggQq00qoGoGggoQOGgGqgQGO0OQ0 atomic_noexcept
#define O0ogoQOgQGG0QO0oGg0oGGGQQOqGgO000 auto
#define OQog0Q0gqgOqOG00ogQoGG0GOGgOGOGo0 bool
#define oqQGqQqoGoqo0qOgoO0gqQog0GOgGQqGq break
#define qG0qqgQoOQ0gQG00qQQQOqQqOGgO0Oqqq case
#define ogOG0qgqoQOGqG0GOoQGo0GOoQGg0GgQo catch
#define Qgqoq0QgO0goQgg0GG0oqOOG0OOgG0QOQ char
#define qO0qQQqQGgqQqOO0goGogq0qGqg0oQqgO char16_t
#define OGqoqqQ0Goo0oQqoQG0QoO0qq0qOq0qOQ char32_t
#define oO0QGgoqOQgGooqoQoqqOqgGgOoqQOOo0 cin
#define g0GG0gGGogo0qG0OGQ0oGog0Q0OQ00GGo class
#define OqoOqgGQ0QqgGQ0oOgQoGoog0g0qQoqQO co_await
#define oOQQ00QGogGqqOGgg0o0goGGQgQQoOggq co_return
#define oOog00q0qOgqOGqgQ0OQ0qo0gQoGGgoGO co_yield
#define ooggQgGQO00goo0oOGQqo0o0O0GqG0GOo concept
#define ogQoOg000oQGQg0QqoqQOGq0oOQGqQOgO const
#define oo00o0g0OOg0Oo0Q0ooGGoooOqqQ00OO0 const_cast
#define qoGoGQqog00ooOqq0GOg00OQgQGOOOOqq constexpr
#define qoGQQgqoQ000OgQGOoQoGGQGqqqQqq0Oo continue
#define gOQOGgg0OoOQOoQOGgg0OoGGg0qQ0QqQQ cout
#define QgO0oQqQgo0Go0QoOG0qqQQG0O0gGO0OG decltype
#define Q0OoQ0qG0QoQq0Q0ooooGQ0gQ0GgGGooq default
#define oQqoQog000G0QQoo0OG0000OqoqGOqOQG delete
#define qQGOQ0oQgOQooGOGQqQ0QgQgo0QggogOq do
```

```
#define QgqgqoGQogQ0g0gGG0ooqgOqqoGqgoOQO double
#define g0oGOqgqqqoGQGg0QO00oQgOOO0ogoGoG dynamic_cast
#define OOqoooo0QOoO0GOQo00oQoqGoQqO0g0gq else
#define ooq0OQogO0GqOGOOoqQ0qoGQ000GgOqoo endl
#define goGOoQooOo0g0qqG0oggGOQ0gQGQg0OqQ enum
#define gGQGQGqqQqg0GG00OoOOQGggO00go0g00 explicit
#define oqQ0Oo0qqQOGQqo0QGGoG0QqQg0OOGgOo export
#define oqO0oqQOO0oqogOOoqQQoOOQgqOGogoo0 extern
#define OqO0gq0qO0ggOOGoGq0OOQoo0GoqQqgoq false
#define qqOqO0gGQ0O00qGgGGG0GgG0OOgG0G0qg final
#define G0goGoGQGO00qoOOGG0GQqGGOqqoogQgO float
#define ogQgOOQQGq00O0QoGGq0og0GGqqgoo0Q0 for
#define OoGqggQOgoogQggOoqGOGOgQoqGgOoGQq friend
#define o0oqggqoO0GQQgOGq00g0Qq0Gq0gGQoQ0 goto
#define OG0gQ0Q0Q0Q00qogoG0o0goqO0oO0qQ0Q if
#define OOqq0goqoqggOggOoq0QOoG0O0GGO0GQq ifstream
#define ogGqOQQQGg0OQOOoG0Oq0qgoQ0Oqq0O0o import
#define q0QoGgoqQgQqQOQgOGO0GoGGQq0QooGoo initializer_list
#define QOooOOOQoqgGgooGG0Q0qgqOOqQ0gOQGO inline
#define GG0q0qgo0O00GqooOO0Q0qqGqOGQOOOGG int
#define QgqGOoOqqQO0GGQ0ogQoQ0qqqOG0oqGOO iterator
#define OoqGQgggGGOQOg0gooOGqo0QQG0O0o0oq list
#define oQGOOOOoG0O00OOqooggGoggQgoGg0OoQ long
#define O00ogqoGogg0g0QooqQO0gQgOqOOQGG0g main
#define oO0oOOQQ0QG0o0GQOGqGg00qGQoqo0GGq map
#define Gg0QgQoQGOOQoQOOGoOqqQgogoqQ0oQq0 max
#define QQOGQ000GQOgOqGq0g0g0QqQqqO0gOgOg min
#define gogoq0oQ0qoOOGQOQ0ogq0QGggQqgGo0G module
#define QQgqo0QgOoQgGqgOO0g00qOGoQ0googGG mt19937
#define qgGgoog0GoqoOo0QqoQqqQG00o00GGOGQ mutable
#define qQo0oGQg0Q00QQOqgG0o00gOOG0gGgqGQ namespace
#define oQ0gQOQQqGOOgOQO0Gg0ogQqooG0O0oqO new
#define G0ooo0oQQOq0OGQ0qGOGoQo0OQqoQGGgg noexcept
#define qOqqggqQoGG0QogogqooOGqgG0qQogGq nullptr
#define ogq0o0oGQgoO0QO0OgoQgqOqG0o0q0OGo ofstream
#define o0GgGoGQOQqQgogQOG0ogGQQo0oOqqqoO operator
#define QOgooogQGO0QGGq0QgoQO0qooQGGQoooQ other
#define qg0oGQQqOQGQqQoO0G0qqqoqQOQQ0QqgG override
#define GGqG00qoGqOOqQOQQGqoqGqGqOQqGgO0o private
#define o0qoggoGqQ0q0oQ0gOgGOgOO00Qgqggqq protected
#define oOOQOQGQoQQ0QQO0qq0GO0QOQOqOGqgoq public
#define Ggq0OqoOQQOOQgG0gGqOOqqGo0Go00Qgo random_device
#define qQ0oq0qGoOQ0q0gQq0oggQgGo0q00GqgG reflexpr
#define OOGGGO0Og0qog0oGOGQGqGGGQ0GG0G00o regex
#define ggQOooGQQ00oGGgq0q0QoOqOQ0GqoqQQ0 register
#define OO0OQGQqGgOggG0g0Qo0Og00goqOG0qgG reinterpret_cast
#define QOGQQo0QoG0OgoOQ0OoQoqq0GGGO00OqG requires
#define oq0qqOOgogOgooqOGGoo0OgqQG0GQq0Gq return
#define oqGq0oQGooGgQq00gqo0oqQ00QgQooQO0 set
#define goGoqq0gqoOGGqOGOQgo0qQQG00G0qqo0 short
```

```
#define QQQo0QOooOgq0GooogQ0oQqq0GOqo0Qg0 signed
#define ooQ00QQo0Qoq0Gq0Q0goG0oOgggOoGoGo size_t
#define QgGQgGgGQo0QqQ0QgggG0OGqQOQqg0go0 sizeof
#define qOqooO0qoqoqQqoqGoOOg0g0GogGq0oQ0 smatch
#define OG0oQQoGQqOgOo0QGqooQGQ0qQqQQ00GG sregex_iterator
#define GQGOGO0o0oGOgQqoOoGqQ0GOgQGQgOoGG static
#define oQggQGoG0g0QQG0OQ00qQ0gGqqQo0Q00g static_assert
#define Q0GGooQQOgOoQGOoogqO0OQqOqGQqQG00 static_cast
#define oGOggOoOQQ0qqqQOGQqQGO00gOQQoQqqO std
#define gQqq0qGqoogOOgoo0OGgG0GOQG0Og0q0Q string
#define ggGQgg0GgqqQgqqo0qGOQogqOQG0gQ0gq struct
#define qOq0Q000oOQgGqoGgoOoGgOG0OqqqQqGQ switch
#define oOgGQq0oGooqG00q0OooQOoooGOGGqOOQ synchronized
#define O0GqoGQqoQgOq0gggoqOQgo0OQOoOqQgg template
#define oGgooQqqog0gqG0GooOgooogGQgOGO0GG this
#define oQOOqQoOg0Qq0gO0G0qogQGqqqqQggOQg thread_local
#define Q0oOGO0Q00Qooq0OogQgG0ggOG0Qg0gGG throw
#define g0oooGoOqqqoqGOoogG00gqoqQ0gOqOQQ transaction_safe
#define QGo0QOOQ0qqQOggqOG0ooOOoOqggQ0oOo transaction_safe_dynamic
#define ggGgGoGq0qo0qGgqQoO0Oo0gg0QqOOOgG true
#define G0OQQqoOOQGGOGo0QooGoGQG0QQo000oG try
#define goqooOGOGGq0QGOOqo0oGOQoqGgq0GOGq typedef
#define gOG0O0GQG0GGgGQoO00oq0oQggOqogGo0 typeid
#define O00QOg0qQQQgO000Q00OgqQG0oggg0OGo typename
#define QGOqoGggQgqGgQoQGoQQoQoO0gggogOQq uniform_int_distribution
#define gGgOOQGoGqGgggqgQGQQQQgOOoog0qOoO union
#define QgQ0QgGqGQ0QGGoQoOqoQGG00Oqgo0oq0 unsigned
#define oGGQgqQOoo0qQG0o0qgoQgGqgGgQOgQ0G using
#define oGg0qQ0GqGgggqOqg0gooQQQggQg0qQgg vector
#define OqQo0GoOOoGOGG00g0Q0ggo0qQo0GgOO virtual
#define oOOoOGq0QooqQog0GgOgQg0qgQQgQGGGq void
#define oGqoQQgQgGg0g0qqggoGGoogOgoQGQqQQ volatile
#define GqGgQqg0Oo0QOOoOGgqggqggOOq00oQGQ wchar_t
#define GgGQgoQGG00G0ooGQq0Q0qgqqoGoGqo00 while
oGGQgqQOoo0qQG0o0qgoQgGqgGgQOgQ0Q qQo0oGQg0Q00QQOqgG0o00gOOG0gGgqGQ
oGOggOoOQQ0qqqQOGQqQGO00gOQQoQqqO;GG0q0qgo0O00GqooOO0Q0qqGqOGQOOOGG
O00ogqoGogg0g0QooqQO0gQgOqOOQGG0g(){ GG0q0qgo0O00GqooOO0Q0qqGqOGQOOOGG a = 5, b =
10, temp; gOQOGgg0OoOQOoQOGgg0OoGGg0qQ0QqQQ << "Before swapping." <<
ooq0OQogO0GqOGOOoqQ0qoGQ000GgOqoo; gOQOGgg0OoOQOoQOGgg0OoGGg0qQ0QqQQ << "a = " << a
<< ", b = " << b << ooq0OQogO0GqOGOOoqQ0qoGQ000GgOqoo; temp = a; a = b; b = temp;
gOQOGgg0OoOQOoQOGgg0OoGGg0qQ0QqQQ << "\nAfter swapping." <<
ooq0OQogO0GqOGOOoqQ0qoGQ000GgOqoo; gOQOGgg0OoOQOoQOGgg0OoGGg0qQ0QqQQ << "a = " << a
<< ", b = " << b << ooq0OQogO0GqOGOOoqQ0qoGQ000GgOqoo;
oq0qqOOgogOgooqOGGoo0OgqQG0GQq0Gq 0;}
```

```
Before swapping.
a = 5, b = 10

After swapping.
a = 10, b = 5


...Program finished with exit code 0
Press ENTER to exit console.
```

Пример работы №2 (таблица умножения):

Оригинальный код:

```
#include <iostream>
using namespace std;
int main(){
    int product;
    int counter=1;
    int number;
    cout<<"Enter the number to print the table: ";
    cin>>number;
    in:
    product = number*counter;
    cout<<number<<" x "<<counter<<" = "<<product<<endl;
    counter = counter + 1;
    if (counter <= 10){
        goto in;
    }
    return 0;
}
```

```
Enter the number to print the table: 11
11 x 1 = 11
11 x 2 = 22
11 x 3 = 33
11 x 4 = 44
11 x 5 = 55
11 x 6 = 66
11 x 7 = 77
11 x 8 = 88
11 x 9 = 99
11 x 10 = 110


...Program finished with exit code 0
Press ENTER to exit console.
```

Измененный код:

```
#include <iostream>
#define QgGOg0oGo0OGQQGQGGo0OoqqqQQoOg0oO abs
```

```
#define qQQ0ogoOOO0QooOQ0GqgGGOO00oOQo0Gg alignas
#define q00gQOogQG0qqoGgq00oqqOQGgogOOqQQ alignof
#define ogqGQOqqo0gQOq0QgOo0oqOQOGOGqqqOg asm
#define oqGGqoOGoqqOOQ0OoQ0O0ogOGGOGQqqo0 atomic_cancel
#define oqOo0qOGQgQOGqqooGO0gGgGooQGQ0o0g atomic_commit
#define oG0GGqgggOoq0GQGggQQQqqOg0GQqO0Gg atomic_noexcept
#define OoOQqGgQQQgooqO0qQOO00qqoqOOgGQOq auto
#define oooqQQOggo0GgG0qqQ0oOGGg0qOg0OoGq bool
#define ogq0gQqQGoGoQ00GQgOoqoOQQOo0qoO00 break
#define oo0gqG0OGoGqGOQogo0Oooq0gGOqogoG case
#define gGgoogogqOqoOOqqQogQOGoOOggOgQogo catch
#define GgGQq0QQGGgqqGgoqgOQQqgo0gqqoogQQ char
#define ggqGQgoggOoOQQgqO00G0qooQQqqqGgQO char16_t
#define O0Goqg0q0GqGgOQOGQgG0gqQQGQGQgGG0 char32_t
#define oOOgoGqoOooGG0oO0q00ggoQO0GoOgQgG cin
#define q0o00Ooqg0GoGoQG0oqO0Qq0Goo0ooqog class
#define gQg0OQqoOqGGQqqgGOGoQ0gqqGggOOgGq co_await
#define GOOqGq0OOOoQGOGqOq0O0oooqQG0QQQG0 co_return
#define QooOgOgq0oQGQgoOqOgGqGo0gGQQOGqoo co_yield
#define ogGOO00GgQgOQGgQOoGQ0Goqog0q0GOgo concept
#define ooqoGggqoQogg0g0qo00OQg00qGooQoQG const
#define GGQOgogGgOGggQQQqogogO0q00gQgqG00 const_cast
#define qgOo0o0gG0g0oqqQoqOGoqQ0qqO0GqqGG constexpr
#define qgGoqgGgO0ooOoGo0ogogOo0GqGGOQooq continue
#define ooOgGoqQQqQQ0gOOQqgGOgg0qoOggQOq0 cout
#define oqQgQqQq00qgQqGGOQOgOqqOGGOOqOG0g decltype
#define QQ0qgQq0OGqgoq0G0GGgQQOqQQogQoOqO default
#define qgooGOqO0gG0G0QQGQQOGOOGQGo0oqOQG delete
#define ooGGqQggOoo00O0OGOQqqgoQgo0GggQqO do
#define OQ0GgQogoo0ogQGOqOgogQg0ggqqgG0qg double
#define ogGqqOoOqgoOoQgqgQOg0ogG0OQGGqogg dynamic_cast
#define OQ0OG00QOGQQqGgOqqGOQQ00qg0OQoOOq else
#define qQoG0ooGO0goqOgoOoGQOgGQqoOqgoGoq endl
#define OQoqqq0ooqQQO0QoO0qgQ0OgOOQq0q0QO enum
#define qoqgGqGq0O0qqqOgoOQog0Q000QqOOgGg explicit
#define oqOGqgQOGOoqQO0g00qG0qQGQQOGgGqGO export
#define QO0OOgGQGoOQoq0ooQ0qGOqo00q00ggqo extern
#define OGqooogOgqQOgoOG000OGqGOgoo0O0QQG false
#define oQGqQ0oOQ0GGGqoGo0qqOooQG0G0qOQgQ final
#define o0g0GqogoOqGQo0gGGoqG0gqqqog0oGQq float
#define Gg0qQgQgoO0oQOoqg0goOg0gogQqOoGgg for
#define oqGOGQgQgqoGQOqGGgG0QQQGg0Q0ogQGq friend
#define o0gogOogGGGQGqgoQG00OgGggOqO0oQo0 goto
#define qoqqgOOqOqOggOqGGOQoq0qqQOQGQgoGQ if
#define qOGGOgqQGQO0ooO0OGO0qgQo0GOOoGq0G ifstream
#define oQO0oogq00Oq0QogOoGG0GG0QOo0q0Qg0 import
#define OoGggGOqgOQgoOOQQ00GgO0QoGgqOqoOG initializer_list
#define qgQqQQGQ0oQooOoq0oG0qoOogQQG0GQq0 inline
#define gg0OQo0OQ0QQOqOQgQOO0o0OGOGQ0Q0QO int
#define QoGq0oQQQqoOOo0oQgGOGgogqO000qOqg iterator
```

```
#define oOq0gqoqgoQo0qQQGgG0OooQQgQQqQOGg list
#define ooooq0gqGgQGG0qqoOOG0gg0gG00qO0QG long
#define oG0ggqoqo0oqoOQqGQqqgG0Qo0Gg0QogO main
#define oG0gO0QQqG0goqqGGQ00G0o0qoGggoGgO map
#define goOOGggqGoqoooooooG0oGQOO0oQ0OoOgG max
#define QOQGgQqog0Q0qoOOqgOgoogqGqq0oQQQQ min
#define oo0OQ0QqggO0qoOqgGqqoqQOOQ0gGOooq module
#define OoQ0G0GoQ0qOgoQQoOGOO0OGOGGqQOgq0 mt19937
#define ooqggQoqOqq00g0OqQQGOgQoq0OooGgoq mutable
#define GGQ0QOgo0Ggqg0GGgOoqQ0qqgGOGGqq0q namespace
#define QGqQOOOoGGOGQo0qOGg0OQ00Q00QQQgOq new
#define qgOOOo0OOqOGqgooGqqQQqOgQQGOq0GQQ noexcept
#define oqqoQQGGOgoQoQg0qGQQGoOQoOgOQO0qO nullptr
#define QoGO0qqooqqGqqqOqGgGqq0qGgqgqoOgg ofstream
#define QqQg00O0Qg0qO0GqoGqo0gG0GOgO0QOGg operator
#define oOog00ooQqgO0oooog0QO0q00gqG0gGG0O other
#define o0g0oQq0QG0Gqoogqg0gQOGoG0GOQgOOQ override
#define qGQOq0gGGg0oOGgGoGo0GOOQ000qq0qgq private
#define q0oQoo00OQOOGG0GGQo0q00QqGOogGOqq protected
#define G0gOQg0qg0gooOOGOog00oQoOO0oOogoO public
#define oqOQgOooggoGOQqG00GqQgGoOgOq0OQqo random_device
#define gqooGOQqo0OqQ0o0oOoQGoGgOOo0oQOQQ reflexpr
#define ggGQq0OqGGOG00oOQQOg0OO00OQqGG0OG regex
#define qGOQqQ0q0ggoOqg0QggOG0GOOoQQqgG00 register
#define Qg0goqoqgG0qgQQ0gGoOGGGo0GQOqoGOg reinterpret_cast
#define G0QOOqOgqoOgOqqOQGoO0ogg0qoQqoOOQ requires
#define oqGGoQQgqOggQqq00GOGqOGQQOQG0GQqq return
#define QqgOOGoGQ0QQOOQqOoGGQO00GQOoqgOqo set
#define GQoGOgGOooggq0G0oOGQOGOQggGoO0qgG short
#define Goo0goqgoGQq0GGOgGOQqgGq00GGoogQO signed
#define q0oGOQQoOQooGQG00QGGOoq0oO0qG0QOg size_t
#define oOoG0QggQgoQ0qOoQQooGoq0o0GQqgQQg sizeof
#define qoQq0QOG00gqOogOGOOqoGGogqGqo0G00 smatch
#define gOqqoQ000g0gG0Oq0GgqG0oqgGggqgqQQ sregex_iterator
#define QoqOQq0OoQ0OOqg0qGGoGgQogOGqOOGOQ static
#define OqOOQQoQoqOOQO000QQgOqOooGQQQGoGO static_assert
#define OoOGgoOg0GGOGqGOOgGGOGgoog0qQqqoO static_cast
#define qG00qq0oOoOgqOQo0Qg0g0G0oOGOoQQQG std
#define qgGgGqGQGOqo0ggqqGqGOQoog0Gg0qGgg string
#define oo0QGOGoQgG00OgogOOG0qqQGgOgGOQqg struct
#define gO0G0QQqQgOoqQqqoqgQQGQOOgq0qQgOg switch
#define QoGQ0OoQgQO0o0g0gQ0G0qO0GQoo0Q0GQ synchronized
#define oogo00OOgOggqQoOGQoQGggo0O0gOG0Og template
#define QOGOQGQgQQQGoq0Gogqgo0QqQgGOqoOGO this
#define goGqOqog0oooQQQGqoOGoGOO0QGOooQOg thread_local
#define oG0GGgQQOG00QoqQ0Q0gqgoqOOO0QgGqq throw
#define o0qQGQQgQOOGO00OoGoQG0qoQ0g0oogq0 transaction_safe
#define qGOgoggqqgqqqgqqoOGqqGoG0q0O0QGog transaction_safe_dynamic
#define ooo0GqOgoGOQGogGOqQqQgQ0Qqo0Qg0qO true
#define oqggOOqqGQoOOGqgGg0O0q0OQOGOOgo0o try
```

```
#define oQG0oOOqgGqG0G0OQGo0QGOGOGqoOqGoq typedef
#define qoq0qqQ0QGoGOGoGgQOGqOQ0qogggGGQQ typeid
#define o0Goqo00QQoGQq0QgG0qq0qG0GGQ0Gogo typename
#define ggqGGqGQqQGOggGOGgOQoqoG00qGooQ0g uniform_int_distribution
#define OQO0GgqG0qQ0qgGoggQ0QQgGoGgOqoGqO union
#define G0qQOgogOOGQGgqQgO0GGGOgoqQ0o0O0O unsigned
#define qOgoQQg0oGg00qgo0oOOGogoqoGog0Ggg using
#define q0QqOqgQ0gooqOOQgO0OGQ0QOq00oggqO vector
#define GoGgO00oO0QQOgoqG0qqoO0gQGgQqOgQO virtual
#define GoOQgGoqoGqgqqqGG0GoGOoogGOQQO000 void
#define OgqqGgO0oQOGGQQGqo0qOggq0Q0qGOoGg volatile
#define G0qoqOgGGqoQ0qOOOqooo0ogGqoQOg0gg wchar_t
#define qGg0GQgoq00gqO0qGqGOqQq0OOoOQQoG0 while
qOgoQQg0oGg00qgo0oOOGogoqoGog0Ggg GGQ0QOgo0Ggqg0GGgOoqQ0qqgGOGGqq0q
qG00qq0oOoOgqOQo0Qg0g0G0oOGOoQQQG;gg0OQo0OQ0QQOqOQgQOO0o0OGOGQ0Q0QO
oG0ggqoqo0oqoOQqGQqqgG0Qo0Gg0QogO(){ gg0OQo0OQ0QQOqOQgQOO0o0OGOGQ0Q0QO
go0Ooq0qQ0QQQQGgq0Qo0ooOQoGg0oO0O; gg0OQo0OQ0QQOqOQgQOO0o0OGOGQ0Q0QO
Og0oqGQqQGQQ0QGOOoo0QGOqQOgQg0QqO=1; gg0OQo0OQ0QQOqOQgQOO0o0OGOGQ0Q0QO number;
ooOgGoqQQqQQ0gOOQqgGOgg0qoOggQOq0<<"Enter the number to print the table: ";
oOOgoGqoOooGG0oO0q00ggoQO0GoOgQgG>>number; in: go0Ooq0qQ0QQQQGgq0Qo0ooOQoGg0oO0O =
number*Og0oqGQqQGQQ0QGOOoo0QGOqQOgQg0QqO;
ooOgGoqQQqQQ0gOOQqgGOgg0qoOggQOq0<<number<<" x
"<<Og0oqGQqQGQQ0QGOOoo0QGOqQOgQg0QqO<<" =
"<<go0Ooq0qQ0QQQQGgq0Qo0ooOQoGg0oO0O<<qQoG0ooGO0goqOgoOoGQOgGQqoOqgoGoq;
Og0oqGQqQGQQ0QGOOoo0QGOqQOgQg0QqO = Og0oqGQqQGQQ0QGOOoo0QGOqQOgQg0QqO + 1;
qoqqgOOqOqOggOqGGOQoq0qqQOQGQgoGQ (Og0oqGQqQGQQ0QGOOoo0QGOqQOgQg0QqO <= 10){
o0gogOogGGGQGqgoQG00OgGggOqO0oQo0 in; } oqGGoQQgqOggQqq00GOGqOGQQOQG0GQqq 0;}
```

```
Enter the number to print the table: 9
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90


...Program finished with exit code 0
Press ENTER to exit console.
```

Пример работы №3 (класс):

Оригинальный код:

```
// Create a Car class with some attributes
class Car {
  public:
    string brand;
```

```
    string model;
    int year;
};

int main() {
  // Create an object of Car
  Car carObj1;
  carObj1.brand = "BMW";
  carObj1.model = "X5";
  carObj1.year = 1999;

  // Create another object of Car
  Car carObj2;
  carObj2.brand = "Ford";
  carObj2.model = "Mustang";
  carObj2.year = 1969;

  // Print attribute values
  cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year << "\n";
  cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year << "\n";
  return 0;
}
```

Измененный код:

```
#define oG00oqqqgo0GGqggQQGO0Q0OqQOOOgqOG abs
#define oOQgOqOGQoOQqQooOQgqQGqQgGqGgOGOq alignas
#define ogq0QoqqOGo0OoQgooggOQOoOoGQ00G0g alignof
#define Oqgg0QqqOGqQgGg0Qgq0GgqoOo0QGgQqO asm
#define OoG0gGOQqgOoQO0o0Q0oqGg00gGq0O0oO atomic_cancel
#define OGGQoGoo0OoQGOQ00OqQgqQqOqqoOqoOg atomic_commit
#define GQqOqOgG0o00O0QQ0GQoqOOGggOqqggoO atomic_noexcept
#define oQqqgOGOogqggoqq0OGgQog0GQqQ0OGgQ auto
#define QO0QqQ0gOGOOoQQGQ0OqGQOQOGgogo0G0 bool
#define OG0gog0GOGOoGGqoQqOq0OqoOggqoOgqO break
#define ooQogqooQQG0oq00QOgOQG0gGGOoOqog0 case
#define oOQqqGQgOGQqOGgGg00OoGqO0g0QgOOGq catch
#define oqgoooOoo0ooQO0qoQ0Ogq0Qqgo0GOOq char
#define Oqq0ooO0G0OGGO000g000Q0GqgGOGggQo char16_t
#define oOg0Og00go0GQqOQGG0ooGqOoOgqqQggG char32_t
#define gGQQgq0gOgoq0qqGqQOg0QGqOOGOOqqGq cin
#define qoOo00OOoOQoQ00q0gOG000gq0gooQqoO class
#define gOgoGqQoggoooOGGOGGoq0QqoOqGgoqGO co_await
#define QqOOQ00gG0OooqgQgG0OQQ0GqQoqGOQqq co_return
#define oGqgQgg0OG0OGqoQgGQg0OOoOQgQGoqQo co_yield
#define OGqoQ0GOqOgQq0oo0GoQggqQqGQqO0Gqg concept
#define ooGOGQqoO000g0OqGQOqgo0GOOq0goGqq const
#define Q0qOQOOgoogg0q0O0Ogqg0OgOgggG0oq0 const_cast
#define gqgOQogQ0qqqQGGGGo0o00OQgOgo0O0gG constexpr
#define oqQgoQooQqOqQOG0OG0GoqooQQOOGqQ0o continue
```

```
#define GG0g0gqqqooqOOq0OoqGGOOQOqO0OOGQQ cout
#define gOOgoGoq0qOqo0GQQOGgoQOOGq0QOgGo0 decltype
#define oQqOOOqqgG0GoGO0Qoq0qQ0QGgQ0gOOQg default
#define gOOGg0OOOg0q000QQoGOgooQOqoOoqGGo delete
#define OggQg00oGgqGO0QggOG0OGGoooGgGgQ00 do
#define oOOGooQQggqGG0gg0o0qqoOOOqQgogqGq double
#define Qg00oqqGQOoqGGQQoOQqQ0GoooOgOQq0q dynamic_cast
#define G0qqG0OOGoGGqO0GqQgGGOqq0QQQqgGqO else
#define qGG0oQGOGQqGO0ggqqGoqgG0QogQqg0Og endl
#define og00oq0ggGqq0QOgQOoGGo0QgOoqQqGQQ enum
#define Oo0OoQOoQgOgQOO0oGgqOOggQQgQqo0qO explicit
#define oQg0ggGGOGOgoQGQQGGGQoOQgQ00QggGq export
#define o0oOOGO0ooOOGQQOoqGqoO00Gg0G0gQqo extern
#define OGqgq0qG0QooQQQqoQqqOoQoqg00goQgO false
#define gQqQgQ0000g0gO0GoOq0GOqQg0qoGO0QO final
#define oGqGooQqQq0OQQoO0G0Q00oo0ooQGqGQQ float
#define qgQg0Q0qq00Qg0g0o0QoOQgGOGogogQGo for
#define oGOggoGQ0Gqgg0oO0o0OGGQ0Q0qOogg0O friend
#define oQ0oOQQQQq00goqQ0gOgq00OQQ00gOGGO goto
#define OOOgGOqGq0oqGqg0oQGGoooogOoq0GggQ if
#define oGoGQgG0QQ0qGoOOQQO0qq0G0GQgOogOo ifstream
#define qOo0OOGg0Q0OQgoO0Og0gQGoGG0qo0qgO import
#define OgogQ0qQgGgO0Oq0gGggGoQooo0Q0gqog initializer_list
#define GQO0OQGQOo0QgGqgoG0qoQoOGQgqgqqG0 inline
#define goqOQ0gogqOgoooQqQOgggqQGQOgQqQqg int
#define qQQqQg0g0GgGqGqqQOQgG0QOgOgQQQQgq iterator
#define oqGGGqg0GOOQOQO0gOgQoQoQo0oO0QOo0 list
#define QO0qg0QgQgq0QGqqQQQQg0gogqOOQQQQg long
#define oGQoQqGO0G000oOgQOoOQ0g0QQGooGQq0 main
#define qGOGOoo0OGGqqO0O0GggOoGgoGqQqgggO map
#define gG0GggQgqGo0oo00oOgq0OQ0OQ00oogoQ max
#define gOoqGGOgQQ0OQGoQq0QoQgoQqQGQoGGoq min
#define gQ00OqgqG0gq0Og000GO0oGGQqO000qOg module
#define oGOOOOq0G0GOoo0qooQqgQ0Og0GQoQ0QO mt19937
#define Go0gQGGqgoQoqqQgqOgooG0OqoOgoo0qq mutable
#define G0G0qQ0oO00QQo0OOOQgoooqOO000QOq namespace
#define QqQ0ggQOOqg0gQq0oOgGGGoOqO0q0Goo0 new
#define OGOOGoQG0GGGGOOQqOG0oqQGOqgQG0qgq noexcept
#define GGQGGoGgQoqQOo0OgG0qqooGGoo0GQGGO nullptr
#define oOqgG0ogOgq0O0GGG0GgQqOOGQoQQQoo0 ofstream
#define og0Gq0GGgqoQgQOGoo0qQoQq0goGGoGqq operator
#define o0goqoQQOOgQ00oGQgOGgqO0GqOoG0QO0 other
#define QgOooq0qo0oGQooOGggGqqogggQ0OGq00 override
#define QgqgGQOQgggoq0oGgOGqgq0QgQQOQOqoq private
#define oogGQGGqg0OgQQOOoo0o0QOOgoG0QoGOO protected
#define oOggQ0qoGQoG0QQGg0Q0qqgQGOG0QOoqq public
#define oGgQ0QOO0ogoOqqQGGgooOG0OOg0goOOG random_device
#define Go0g0oGOOOoQoQ0gO00gqQqoqOgQQoqGg reflexpr
#define qqQqo0Oo0OG00oqO0goGgqoooGGOQQqoq regex
#define oOgqogOOQgGQ0oG0GGgOQQQQQqGgOO0o0 register
```

```
#define OGqQOqggoOgQQo00qQqqqOoqGQgOOGgoO reinterpret_cast
#define gqGQOgoOQGOqQOOgqQ0GQOQOOqogOoqqo requires
#define oOOOOOOqqOqOoqGGqogQ00qOOQQg0O0qq return
#define oqGOg0Gog0QGOOoqQO0O0qQGgOQOq0gqq set
#define GOgO0o0OO0GG0oo00G0Gg0gqGQOQGOQQG short
#define oGgOoO000q0qoQOOQOqgOOGGGoOGGqogq signed
#define oGoooO00gqgQGoOqqoooOooGqooGGQGO size_t
#define Q0qQQ0GqGQggGgqG0OogG0ggQOo0GQq00 sizeof
#define qgGooGQoQGqqoqqqgg0OQ0Ogg00goQgoq smatch
#define G0OOqGogooGGgqgo00gG0GqoQGOOOQoQO sregex_iterator
#define gQ00qQOQ0GGoqGGQqGqQ0GOGQ0oOO0QOQ static
#define qqGGGG0O0oogqQ0ggOq0qqGGq0OQQqGO0 static_assert
#define oGGgQOoo0qGqggQOgQqo0g00QOG0qGqgG static_cast
#define oQoGQ0QoGQqqQ0ggoGg0qoGOOQOgqoqoq std
#define o00qGQQOoGooOGOqQ000gq00OqQ0OgOQO string
#define o00ooooqOgo0oQOqqoQooqQGqggg0qOQO struct
#define gqgQOog0ooQqogg0O00OqgqgqGGGg0GGQ switch
#define oGGo00o0QO0OqOGqO0QGO0gOqOQQOOOoG synchronized
#define QoqOOqqoOO0QQGqqo0OGGGqgOGoGOqGog template
#define QoGoqoOGqqGggoOGoQoqq0oGQ0gg0GOGQ this
#define QQooOQoGQQgqgGogQQgGqGQgoQqggooqQ thread_local
#define oOQq0O0GOGGooQGQgqQoqogoQoGOoQOQG throw
#define qq0qg0oqQQ0G0gQOgQqg0QoO0oQQqgOqG transaction_safe
#define gQoQg0qOgq0GQo0g0GgOgg00oQQqGQgoG transaction_safe_dynamic
#define OoOg0g0QO0Q0QqQOqoGgQoq0OQOOQgoOG true
#define gGOGoqgQqqqGqqGgOooqQ0OoOOggQOog try
#define GqoQg0OQoo0OgOqQqqqgggq0GgQq0gO0g typedef
#define qqOO0qqO0QQGg0Oo0GqqQOoQGgG0OoGqq typeid
#define qoqq0GOGOQgqoOGgGoOOQGqoGOGQQGogg typename
#define Qo00OGgqG0QGgqQgqQoqoqO0oQgggOgG0 uniform_int_distribution
#define oQo0Og0QGqQqogq0gqgQq0GOgq00qOoOG union
#define G0oQo0gqqOOo0gO0oOGoOggqgg0G0GoQg unsigned
#define GoQO0oOogO0gQQQOOQGOoqo00QOq0QQqQ using
#define QOQ0q0g0oQGog000Qo0ggOGoGo0OGgoGg vector
#define oqOgoqoGGQgO0gOQqqgg0gOGGoggGqooo virtual
#define ggQ0OGoOqOgGoqgoqgoQqQGOOG0ggG0Qq void
#define ooOoo0QQgQgQoQg0qqogoogOgqG0g0goQ volatile
#define GOQq0gGG00OOoogQQOGOQogggoqqgG0go wchar_t
#define og0qOo0oQ0qGOgGQGQoOgGqq0oqgOoQGq while
qoOo00OOoOQoQ00q0gOG000gq0gooQqoO Car { oOggQ0qoGQoG0QQGg0Q0qqgQGOG0QOoqq:
o00qGQQOoGooOGOqQ000gq00OqQ0OgOQO QQ0Qg0GqggQoGgGgQ0OgogqQGOGq0Qqqo;
o00qGQQOoGooOGOqQ000gq00OqQ0OgOQO qGGOGQ0OGg0gGOQqoGgGo0oQoqQqOo00q;
goqOQ0gogqOgoooQqQOgggqQGQOgQqQqg
g0o0oqgGGQqGGQOQOOgggggqg0QOqO0qo;};goqOQ0gogqOgoooQqQOgggqQGQOgQqQqg
oGQoQqGO0G000oOgQOoOQ0g0QQGooGQq0() { Car carObj1;
carObj1.QQ0Qg0GqggQoGgGgQ0OgogqQGOGq0Qqqo = "BMW";
carObj1.QGqggGQGOGqgOQGogqoo00GQGGO0GOGQ = "X5";
carObj1.g0o0oqgGGQqGGQOQOOgggggqg0QOqO0qo = 1999; Car carObj2;
carObj2.QQ0Qg0GqggQoGgGgQ0OgogqQGOGq0Qqqo = "Ford";
carObj2.QGqggGQGOGqgOQGogqoo00GQGGO0GOGQ = "Mustang";
```

```
carObj2.g0o0oqgGGQqGGQOQOOgggggqg0QOqO0qo = 1969; GG0g0gqqqooqOOq0OoqGGOOQOqO0OOGQQ
<< carObj1.QQ0Qg0GqggQoGgGgQ0OgogqQGOGq0Qqqo << " " <<
carObj1.QGqggGQGOGqgOQGogqoo00GQGGO0GOGQ << " " <<
carObj1.g0o0oqgGGQqGGQOQOOgggggqg0QOqO0qo << "\n"; GG0g0gqqqooqOOq0OoqGGOOQOqO0OOGQQ
<< carObj2.QQ0Qg0GqggQoGgGgQ0OgogqQGOGq0Qqqo << " " <<
carObj2.QGqggGQGOGqgOQGogqoo00GQGGO0GOGQ << " " <<
carObj2.g0o0oqgGGQqGGQOQOOgggggqg0QOqO0qo << "\n"; oOOOOOOqqOqOoqGGqogQ00qOOQQg0O0qq
0;}
```

Пример работы №4 (файл библиотеки):

Оригинальный код:

```
#include "binMatrix.h"

binMatrix::binMatrix(std::initializer_list<std::initializer_list<bool>> list)
{
	for (auto& i : list)
	{
		mtr.push_back(std::vector<bool>(i));
	}
}

std::ostream& operator<<(std::ostream& out, const binMatrix& mtr)
{
	for (int i = 0; i < mtr.mtr.size(); i++)
	{
		for (int j = 0; j < mtr.mtr[i].size(); j++)
		{
			out << mtr.mtr[i][j] << " ";
		}
		out << std::endl;
	}
	return out;
}

binMatrix binMatrix::operator*(const binMatrix& other)
{
	binMatrix result;
	for (int i = 0; i < mtr.size(); i++)
	{
		std::vector<bool> temp;
		for (int j = 0; j < other.mtr[i].size(); j++)
		{
			bool sum = 0;
			for (int k = 0; k < mtr[i].size(); k++)
			{
				sum += mtr[i][k] * other.mtr[k][j];
			}
			temp.push_back(sum);
		}
```

```
		result.mtr.push_back(temp);
	}
	return result;
}

binMatrix& binMatrix::operator*=(const binMatrix& other)
{
	binMatrix result;
	for (int i = 0; i < mtr.size(); i++)
	{
		std::vector<bool> temp;
		for (int j = 0; j < other.mtr[i].size(); j++)
		{
			bool sum = 0;
			for (int k = 0; k < mtr[i].size(); k++)
			{
				sum += mtr[i][k] * other.mtr[k][j];
			}
			temp.push_back(sum);
		}
		result.mtr.push_back(temp);
	}
	mtr = result.mtr;
	return *this;
}

bool binMatrix::operator==(const binMatrix& other)
{
	if (mtr.size() != other.mtr.size())
	{
		return false;
	}
	for (int i = 0; i < mtr.size(); i++)
	{
		if (mtr[i].size() != other.mtr[i].size())
		{
			return false;
		}
		for (int j = 0; j < mtr[i].size(); j++)
		{
			if (mtr[i][j] != other.mtr[i][j])
			{
				return false;
			}
		}
	}
	return true;
}

binMatrix binMatrix::operator-(const binMatrix& other)
```

```
{
	binMatrix result;
	for (int i = 0; i < mtr.size(); i++)
	{
		std::vector<bool> temp;
		for (int j = 0; j < mtr[i].size(); j++)
		{
			temp.push_back(mtr[i][j] && !other.mtr[i][j]);
		}
		result.mtr.push_back(temp);
	}
	return result;
}

binMatrix& binMatrix::operator-=(const binMatrix& other)
{
	binMatrix result;
	for (int i = 0; i < mtr.size(); i++)
	{
		std::vector<bool> temp;
		for (int j = 0; j < mtr[i].size(); j++)
		{
			temp.push_back(mtr[i][j] && !other.mtr[i][j]);
		}
		result.mtr.push_back(temp);
	}
	mtr = result.mtr;
	return *this;
}

binMatrix binMatrix::operator|(const binMatrix& other)
{
	binMatrix result;
	for (int i = 0; i < mtr.size(); i++)
	{
		std::vector<bool> temp;
		for (int j = 0; j < mtr[i].size(); j++)
		{
			temp.push_back(mtr[i][j] || other.mtr[i][j]);
		}
		result.mtr.push_back(temp);
	}
	return result;
}

binMatrix& binMatrix::operator|=(const binMatrix& other)
{
	binMatrix result;
	for (int i = 0; i < mtr.size(); i++)
	{
```

```
            std::vector<bool> temp;
            for (int j = 0; j < mtr[i].size(); j++)
            {
                  temp.push_back(mtr[i][j] || other.mtr[i][j]);
            }
            result.mtr.push_back(temp);
      }
      mtr = result.mtr;
      return *this;
}

binMatrix binMatrix::operator&(const binMatrix& other)
{
      binMatrix result;
      for (int i = 0; i < mtr.size(); i++)
      {
            std::vector<bool> temp;
            for (int j = 0; j < mtr[i].size(); j++)
            {
                  temp.push_back(mtr[i][j] && other.mtr[i][j]);
            }
            result.mtr.push_back(temp);
      }
      return result;
}

binMatrix& binMatrix::operator&=(const binMatrix& other)
{
      binMatrix result;
      for (int i = 0; i < mtr.size(); i++)
      {
            std::vector<bool> temp;
            for (int j = 0; j < mtr[i].size(); j++)
            {
                  temp.push_back(mtr[i][j] && other.mtr[i][j]);
            }
            result.mtr.push_back(temp);
      }
      mtr = result.mtr;
      return *this;
}

binMatrix binMatrix::operator^(const binMatrix& other)
{
      binMatrix result;
      for (int i = 0; i < mtr.size(); i++)
      {
            std::vector<bool> temp;
            for (int j = 0; j < mtr[i].size(); j++)
            {
```

```
                    temp.push_back(mtr[i][j] ^ other.mtr[i][j]);
            }
            result.mtr.push_back(temp);
      }
      return result;
}

binMatrix& binMatrix::operator^=(const binMatrix& other)
{
      binMatrix result;
      for (int i = 0; i < mtr.size(); i++)
      {
            std::vector<bool> temp;
            for (int j = 0; j < mtr[i].size(); j++)
            {
                    temp.push_back(mtr[i][j] ^ other.mtr[i][j]);
            }
            result.mtr.push_back(temp);
      }
      mtr = result.mtr;
      return *this;
}

binMatrix binMatrix::operator~()
{
      binMatrix result;
      for (int i = 0; i < mtr.size(); i++)
      {
            std::vector<bool> temp;
            for (int j = 0; j < mtr[i].size(); j++)
            {
                    temp.push_back(!mtr[i][j]);
            }
            result.mtr.push_back(temp);
      }
      return result;
}

binMatrix binMatrix::operator!()
{
      binMatrix result;
      for (int i = 0; i < mtr.size(); i++)
      {
            std::vector<bool> temp;
            for (int j = 0; j < mtr[i].size(); j++)
            {
                    temp.push_back(mtr[j][i] == 1);
            }
            result.mtr.push_back(temp);
      }
```

```
	result &= *this;
	return result;
}

binMatrix binMatrix::identity()
{
	binMatrix result;
	for (int i = 0; i < mtr.size(); i++)
	{
		std::vector<bool> temp;
		for (int j = 0; j < mtr[i].size(); j++)
		{
			if (i == j)
			{
				temp.push_back(1);
			}
			else
			{
				temp.push_back(0);
			}
		}
		result.mtr.push_back(temp);
	}
	return result;
}

binMatrix binMatrix::pow(int n)
{
	binMatrix result;
	if (n == 0)
	{
		result = identity();
	}
	else if (n == 1)
	{
		result = *this;
	}
	else
	{
		result = *this;
		for (int i = 1; i < n; i++)
		{
			result *= *this;
		}
	}
	return result;
}

binMatrix binMatrix::empty()
{
```

```
	binMatrix result;
	for (size_t i = 0; i < mtr.size(); i++)
	{
		std::vector<bool> temp;
		for (size_t j = 0; j < mtr.size(); j++)
		{
			temp.push_back(0);
		}
		result.mtr.push_back(temp);
	}
	return result;
}

binMatrix binMatrix::full()
{
	binMatrix result;
	for (size_t i = 0; i < mtr.size(); i++)
	{
		std::vector<bool> temp;
		for (size_t j = 0; j < mtr.size(); j++)
		{
			temp.push_back(1);
		}
		result.mtr.push_back(temp);
	}
	return result;
}

bool binMatrix::isReflexive()
{
	for (size_t i = 0; i < mtr.size(); i++)
	{
		if (!mtr[i][i])
		{
			return false;
		}
	}
	return true;
}

bool binMatrix::isAntiReflexive()
{
	for (size_t i = 0; i < mtr.size(); i++)
	{
		if (mtr[i][i])
		{
			return false;
		}
	}
	return true;
```

```
}

bool binMatrix::isSymmetric()
{
	return (*this == (!*this));
}

bool binMatrix::isAntiSymmetric()
{
	return ((*this & !*this) == identity());
}

bool binMatrix::isTransitive()
{
	return ((*this * *this) == *this);
}

bool binMatrix::isEmpty()
{
	for (size_t i = 0; i < mtr.size(); i++)
	{
		for (size_t j = 0; j < mtr.size(); j++)
		{
			if (i != j && mtr[i][j])
			{
				return false;
			}
		}
	}
	return true;
}

bool binMatrix::isAntiTransitive()
{
	return (((*this * *this) & *this) == empty());
}

bool binMatrix::isFull()
{
	return ((*this | identity() | !*this) == full());
}

bool binMatrix::isAsymmetric()
{
	return !(*this == (!*this));
}

bool binMatrix::isTolerant()
{
	return (isReflexive() && isSymmetric());
```

```
}

bool binMatrix::isEquivalent()
{
	return (isReflexive() && isSymmetric() && isTransitive());
}

bool binMatrix::isOrder()
{
	return (isAntiSymmetric() && isTransitive());
}

bool binMatrix::isWeakOrder()
{
	return (isOrder() && isReflexive());
}

bool binMatrix::isStrictOrder()
{
	return (isOrder() && isAntiReflexive());
}

bool binMatrix::isLinearOrder()
{
	return (isOrder() && isFull());
}

bool binMatrix::isWeakLinearOrder()
{
	return (isWeakOrder() && isFull());
}

bool binMatrix::isStrictLinearOrder()
{
	return (isStrictOrder() && isFull());
}
```

Измененный код:

```
#include "binMatrix.h"
#define qGqOG0GQGOgOqg0OOooQqgGgGOQoqQQog abs
#define QG0goGooG00gqQqOqgo0OGqGgQoqGgQgO alignas
#define oQOQoQOgQoOgoqG0QGOGqQQq0oqGQ0qgg alignof
#define oGGg0oOQqO0go0o00G0oqg0Oq0gogqo0o asm
#define oQ00oQGqqQooQQGGgOoqOOgoqoQQg0ogO atomic_cancel
#define gqoOgG0gOog00OQGgqQgGQOgOGOggqg0o atomic_commit
#define gqOQGGQQoq0ooOOOOQgqqqOOQgqgOgG0q atomic_noexcept
#define GqGQqQGGqqQQQ0GQoOOQGQGg0qogGOQgg auto
#define oOoOGOgGogOoGGOggg0QGqogQ00GogqQO bool
#define oOqogOogqqqo0OgQQ0ogGGgoQOQo0ooOg break
```

```
#define OqgGQGqoOQG00OgGOGqGogQQQ0QgGoqoo case
#define QQOoo0OOGoGOq0gOg0GQQQ0qQOGooO0oo catch
#define q0qGGg0g0QOooQ0OOgqOQgGgOO0gQGQQG char
#define goQ0qggoqQgQGQGOGgQqQQ0QG0gGgoqq char16_t
#define ooo0QGGggoOgqoGqGoQogq0g0QqQ0q00g char32_t
#define QG0OGQQqGQoOgogQOoQoqqgGOQOoOGqgO cin
#define qOQoGGGG0qGQOgqGGqqqGgOo0QGqQQooq class
#define Ogogqg0OgqqO0OgqgQoO0oOGo0Q0gggGq co_await
#define GOqG00gGQggQGqoQQQo0O00Go0QqQgGQQ co_return
#define oGggOO0GQqoOOGQOggoQGGGgogGog000q co_yield
#define GG0gOGOggooGqoQG0G000g00o0000QQ0Q concept
#define GOQ0OOQoQQQggGQQQQGqqqQgqgQQoooGG const
#define oOQQQoQOGgQG000QooGGQoqooQO0oOgoG const_cast
#define GGOqG0gGOQOGOg0OQGoQq0GoGoqgqOqQO constexpr
#define Oo0gGGgQQ0qgQ0GQoqgO0QgoQogOGgO0Q continue
#define ooqqG0gQqgQOgQoGO000qqgOQoGQo0QQO cout
#define OGG0Q000Q00OqQoO0qQGgQOooqO0gOO0o decltype
#define qOqQgqq000Oqg0ogoOOgooQOqG0qQg0gG default
#define qQGQqOoqq0qGggoqOqQ0oOQ0QqQGoGgGQ delete
#define oqOOQQOq0oqOGGQOGGGGoOgGqg00oQ0qO do
#define oqGgQgGOOQg0Oqo00GggG0GQ0O0gQoGgG double
#define OoogOOOoggqqqqqGO0Gg0ogqogGo0qOg0 dynamic_cast
#define QGgqqgGqQq0OQqQoGGGg0OooGOoGOOQO0 else
#define GQqGQgoGgooQgQGOGQgqogqOOq0QoqqOo endl
#define QgOogoQgoOqGOq0o0GqGog0GG0OQ0QgoG enum
#define oQGq0GGQoGOOoOGqOgg0Qog0oqgqGo0gO explicit
#define GgggggOGQQ0qqgqgGogoo0GgOgoqOOggG export
#define GG0o0QOQOqOGogqqgGg0OGqooOOOOGo0o extern
#define Ooooq0G0Q00qoqOq0GooGOqqOqOOQQqqo false
#define OGQgq0gOo0Gq0QOG00GoQOqQOggQgGGoG final
#define ggq0qo0qqQ0GoOQQOGGqogGgogO0GQQGO float
#define Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg for
#define ogqg0qOoGoOGGo0G0OOGGGoooGqqOGG0g friend
#define ogqGqOqgoGgQoogoQqgGQgqqggOqOoogQ goto
#define GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og if
#define oOO0qQgqgGq00qqOgoO0OQQoQqoOOQGoO ifstream
#define GGqqqQ0ggqoGOqoG0o0QQGQGQgO0g00Og import
#define O0OQG0qoQOog0oOoOgqqqOo00QGg0qOOq initializer_list
#define oqoogOq0OoOOQOOQo0gogogogogG0qGO0 inline
#define oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG int
#define oooOQGgoQGGGO0gOQ00qOQOoOQG0qqQg0 iterator
#define OgQOg0gqGoGGoGogg0QqGQoOGGQoO0qoQ list
#define qOqG0OgoOGGGQgqQGGGqQq0ggqGgQqooG long
#define QoQogGO0Q00QGGgOQG00Oo0oQQqQqqOQ0 main
#define O0oQqo0qq00gOgqgg00OGQGoOgO0gg0qg map
#define o0Q0Q0oQGQqGoGOoqQgo0o00ogO00gG0g max
#define OQQo0qg0oQQQggqoqgGggGO0GoQOOQOgg min
#define Ooooog0gqoGQGOoOgQq0GGGqoQG0G0GOo0 module
#define QQGQOqOGgQgg0QooO0gOGQOogGQ0GOgqg mt19937
#define o000qGOQQGoqGqggo0goqQ0q0gOgoQgGq mutable
```

```
#define oQg0g00OOogogOoG0QGQqqQgooOqqoOqq namespace
#define gGQqOoQgQqgqq0QggQogOOoQoGOogGqOq new
#define qGogGOOo0qOqOgqq0O0QgQ0Q0gGoG0oGo noexcept
#define gGqgGqG0QQoogGG0oqQ0QoGO0oqQogG0O nullptr
#define ggGoQqo0o00GgoqoGGqoo0OGooooOoo0q ofstream
#define oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo operator
#define OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ other
#define g0OgoQOOQgOqoGg0qqgGO00qQo0o0QgGq override
#define gOg0QGo0gQqGqooooGggq0GgGqQoOooG0 private
#define gqG0qqOgg0OgoGQGqgoQG00GgOQGOgqqQ protected
#define gqGQQo0GqQqoGQQGOqgqqGoQgGQQOQgqq public
#define ooOoQGOqQ0GoqgOoooqGGQgGGQGqO0q0o random_device
#define gQoOoq0Qgo0og0q0Qq0qOq0GQgOgQQG0O reflexpr
#define gQqOGg0qgqoGgqOGgo00ooGgqoGOgQoQo regex
#define GOGO0q0OGgg0OoO0qQgGQOgqo0ooqoO0O register
#define ogqOG0ogQgoGooQG0g0oqoQQqooGOoGqo reinterpret_cast
#define gOOQgqOOQOqGQOo0qQGOgQOgOqoQ0qOQo requires
#define oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo return
#define ooQgQgo0qG0QQo0GOOGoq0G0oOGgOOQgg set
#define gOQog0gQOQg0GgqGoqOOqqg0OOG0gqQQG short
#define gOggoGGG0ooq0OgGg0q00GqO0q0OQoOqq signed
#define o00QogQGqGogQG0Qg00oqqQGOq0GOggoq size_t
#define QG0ggOqOGqOGQQOqqogOoOg0GoGoOQOgG sizeof
#define OQoQ0QqQog0O0QgqGgogqgq00gOOgOOGg smatch
#define QOOOGGGoGQqqo0qOOOgO0GO0OGQQqqoqq sregex_iterator
#define OGogOggq0GOOgqQQgq0gQOgooggoqqo0q static
#define OqqQG0q0ogO00gggqGQQoQOGOO0oQqGog static_assert
#define qoqQo0gqqggGGOgoOOOOQOO00OGGgOQqo static_cast
#define OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg std
#define qGqqqq0GGGoQQgQQoOQooGG0ggggqGOGO string
#define gggG0GQqggOGQQO0g0oGgQQ0GqgqO00GG struct
#define Q0oogqG0qOoOGoO0q00G0G0ogqgG0QOoO switch
#define oG0OQog0oqOG0gOGOOgQGO0GgoG0GGo0O synchronized
#define oQGggOGoQOOGqGQOo0QggQgggg0QQg0Gg template
#define Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg this
#define gO0GOQOOggQQQOqGQg0QoGGO00qGQqO0Q thread_local
#define qg0GGOqG0Qo0ogQqOOGO0Qoq0OooqoQgq throw
#define oGqoqQqOG0oqOOoooGGGGgoggg0G0qGgo transaction_safe
#define GOG0Gq0o0qgqoQgQGQG0Q0qgoogG0oOOo transaction_safe_dynamic
#define qOoG0oq0OGggqGgOqGQQqqOO00OGOGQoo true
#define gq0qggGqqGg0QggQOOoOgqOGGGqgGggg0 try
#define oqoO0g0OqggqgOqQOOQqoqggQ0GGg0OOG typedef
#define oGo00oOoQOqGQ0qqO0GqQGQQOgo0GO0gq typeid
#define oQGQogGQgOG0gqgOOogqGooqqQoqGO0QG typename
#define q00Q0Gq0qGGoQO0QQQoQGgQ0OoOoGGQqq uniform_int_distribution
#define oQQ0gOqGoqoqQQGG0GQoOgoQQG0OqoGg0 union
#define QqOQ0QQoqOoOqoGgQGG0gOGogQ0qGqq0q unsigned
#define GQoOGo0GOG00OGggq0oGogqOg0OgqGQ0o using
#define gggGO0OqGoogggqQQqqqQGo0OGOQG0Gqq vector
#define QgogQ00Go0oOoQgoqggoGGGgqGo0QQGGO virtual
```

```
#define oGogo0GQqqgGGqOG00OqOQ0ogGo0qqqGG void
#define OQO0gOGqOQ0qgogqqOQgGoggGO0GOOGgo volatile
#define GogqogQ0gogGgOoOQoOoQ0qOGo0QgOgQQ wchar_t
#define OQGqGQoo0OqoOG0qGGo00gQQq0q0oOq0q while
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG(OgGqQ00OOO0oqq0
o0OG0QqqOOoqOgqGQg::O0OQG0qoQOog0oOoOgqqqOo00QGg0qOOq<OgGqQ00OOO0oqq0o0OG0QqqOOoqOgq
GQg::O0OQG0qoQOog0oOoOgqqqOo00QGg0qOOq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>>
OgQOg0gqGoGGoGogg0QqGQoOGGQoO0qoQ){Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(GqGQqQGGqqQQQ0GQoOOQGQGg0qogGOQgg& OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo :
OgQOg0gqGoGGoGogg0QqGQoOGGQoO0qoQ){mtr.push_back(OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::
gggGO0OqGoogggqQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>(OOqqg0gQqOGGO0G
0oOGgGq00g00O0qQqo));}}OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::ostream&
oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo<<(OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::ostream& out,
GOQ0OOQoQQQggGQQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
mtr){Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0;
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){out <<
mtr.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] << "
";}out <<
OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::GQqGQgoGgooQgQGOGQgqogqOOq0QoqqOo;}oG0Q0ggOQooo0Q
qQQQo0o0g0o0gQgooQo out;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo*(GOQ0OOQoQQQggG
QQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
qgGQGgq0Q0qOqOOQQGqo0qgQGGGgQq0Og = 0;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG qOQoogQQgG0Qgo00GgGgQQgGOOqGgg0oq = 0;
GqoGOgQ0OGogqQg00ogGOgGg0gO0q0qQO < mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
GqoGOgQ0OGogqQg00ogGOgGg0gO0q0qQO++){oqoOQQ0QG0qOOGQ0gQQQOGGQo0OGoQ00q +=
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][GqoGOgQ0OGogqQg00ogGOgGg0gO0q0qQO] *
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[GqoGOgQ0OGogqQg00ogGOgGg0gO0q0qQO][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q];}temp.push_back(oqoOQQ0QG0qOOGQ0gQQQOGGQo0OGoQ00q);}result.m
tr.push_back(temp);}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo*=(GOQ0OOQoQQQgg
GQQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
```

```
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
qgGQGgq0Q0qOqOOQQGqo0qgQGGGgQq0Og = 0;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG qOQoogQQgG0Qgo00GgGgQQgGOOqGgg0oq = 0;
GqoGOgQ0OGogqQg00ogGOgGg0gO0q0qQO < mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
GqoGOgQ0OGogqQg00ogGOgGg0gO0q0qQO++){oqoOQQ0QG0qOOGQ0gQQQOGGQo0OGoQ00q +=
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][GqoGOgQ0OGogqQg00ogGOgGg0gO0q0qQO] *
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[GqoGOgQ0OGogqQg00ogGOgGg0gO0q0qQO][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q];}temp.push_back(oqoOQQ0QG0qOOGQ0gQQQOGGQo0OGoQ00q);}result.m
tr.push_back(temp);}mtr = result.mtr;oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg;}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo==(GOQ0OOQoQQQgg
GQQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og (mtr.size() !=
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr.size()){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
Ooooq0G0Q00qoqOq0GooGOqqOqOOQQqqo;}Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0;
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo < mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og
(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size() !=
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size()){oG0
Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
Ooooq0G0Q00qoqOq0GooGOqqOqOOQQqqo;}Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0;
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q < mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og
(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] !=
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q]){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
Ooooq0G0Q00qoqOq0GooGOqqOqOOQQqqo;}}}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
qOoG0oq0OGggqGgOqGQQqqOO00OGOGQoo;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo-
(GOQ0OOQoQQQggGQQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O
```

```
0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] &&
!OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qo
qgOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}oG0Q0ggOQooo0QqQQQo0o0g0o0gQg
ooQo result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo-
=(GOQ0OOQoQQQggGQQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O
0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] &&
!OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qo
qgOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}mtr =
result.mtr;oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo|(GOQ0OOQoQQQggG
QQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O
0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] ||
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgo
oQo result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo|=(GOQ0OOQoQQQgg
GQQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O
0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] ||
```

```
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}mtr =
result.mtr;oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo&(GOQ0OOQoQQQggG
QQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O
0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] &&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgo
oQo result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo&=(GOQ0OOQoQQQgg
GQQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O
0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] &&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}mtr =
result.mtr;oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo^(GOQ0OOQoQQQggG
QQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O
0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] ^
```

```
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgo
oQo result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo^=(GOQ0OOQoQQQgg
GQQQQGqqqQgqgQQoooGG OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG&
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O
0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q] ^
OgqOO0qGoqq0Q0gogOOg0G0OoQooOq0gQ.mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0Qoq
gOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}mtr =
result.mtr;oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo~(){OQoQoG0OgQoO
qOGQ0q0oGo0GoOqOQQogG result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0;
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo < mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(!mtr[OOqqg0gQqOGGO0G0oOGgGq00g00
O0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q]);}result.mtr.push_back(temp);}oG0Q0ggOQoo
o0QqQQQo0o0g0o0gQgooQo result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::oqQGGoGggg0qOQQ0GqOogG0Og0oGgO0Oo!(){OQoQoG0OgQoO
qOGQ0q0oGo0GoOqOQQogG result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0;
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo < mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(mtr[oooOGQQ0QoqgOogGGO00O0o0OoGg
GG00Q][OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo] == 1);}result.mtr.push_back(temp);}result
&= *Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg;oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::identity(){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
```

```
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
Q00G0qQqqoooooOoQo0OGQ0GoqGggqQqq = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo].size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og
(OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo ==
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q){temp.push_back(1);}QGgqqgGqQq0OQqQoGGGg0OooGOoGOO
QO0{temp.push_back(0);}}result.mtr.push_back(temp);}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQ
o result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::pow(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG
oQ0gOoqOQqGGGq0OqGQGoQggoG0Oqqoqg){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og (oQ0gOoqOQqGGGq0OqGQGoQggoG0Oqqoqg ==
0){result = identity();}QGgqqgGqQq0OQqQoGGGg0OooGOoGOOQO0
GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og (oQ0gOoqOQqGGGq0OqGQGoQggoG0Oqqoqg == 1){result =
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0Qgg;}QGgqqgGqQq0OQqQoGGGg0OooGOoGOOQO0{result =
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0Qgg;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(oOqqQQQ0QoQGGqOGq00OGoqOQGGgg0ggG Q000G0ooG0QQOoo0Qogq0G0QGggqgoOgq = 1;
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo < oQ0gOoqOQqGGGq0OqGQGoQggoG0Oqqoqg;
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){result *=
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0Qgg;}}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::empty(){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (o00QogQGqGogQG0Qg00oqqQGOq0GOggoq
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (o00QogQGqGogQG0Qg00oqqQGOq0GOggoq
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr.size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(0);}result.mtr.push_back(temp);}
oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo result;}OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG::full(){OQoQoG0OgQoOqOGQ0q0oGo0GoOqOQQogG
result;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (o00QogQGqGogQG0Qg00oqqQGOq0GOggoq
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo = 0; OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo <
mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){OgGqQ00OOO0oqq0o0OG0QqqOOoqOgqGQg::gggGO0OqGoog
gqgQQqqqQGo0OGOQG0Gqq<oOoOGOgGogOoGGOggg0QGqogQ00GogqQO>
temp;Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg (o00QogQGqGogQG0Qg00oqqQGOq0GOggoq
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q = 0; oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q <
mtr.size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){temp.push_back(1);}result.mtr.push_back(temp);}
oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo result;}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isReflexive(){Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(o00QogQGqGogQG0Qg00oqqQGOq0GOggoq OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo = 0;
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo < mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og
(!mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo]){oG0Q0gg
OQooo0QqQQQo0o0g0o0gQgooQo
Ooooq0G0Q00qoqOq0GooGOqqOqOOQQqqo;}}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
```

```
qOoG0oq0OGggqGgOqGQQqqOO00OGOGQoo;}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isAntiReflexive(){Ogq00qoqggQoqQqGGGggOooQOQ0goG0
gg (o00QogQGqGogQG0Qg00oqqQGOq0GOggoq OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo = 0;
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo < mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og
(mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo]){oG0Q0ggO
Qooo0QqQQQo0o0g0o0gQgooQo
Ooooq0G0Q00qoqOq0GooGOqqOqOOQQqqo;}}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
qOoG0oq0OGggqGgOqGQQqqOO00OGOGQoo;}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isSymmetric(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
(*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg ==
(!*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg));}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isAntiSymmetric(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgoo
Qo ((*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg & !*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg) ==
identity());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isTransitive(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
((*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg * *Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg) ==
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg);}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isEmpty(){Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(o00QogQGqGogQG0Qg00oqqQGOq0GOggoq OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo = 0;
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo < mtr.size();
OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo++){Ogq00qoqggQoqQqGGGggOooQOQ0goG0gg
(o00QogQGqGogQG0Qg00oqqQGOq0GOggoq oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q = 0;
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q < mtr.size();
oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q++){GO0gQO0qggGooOo0ooo0ooGG0Qgqq00og
(OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo != oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q &&
mtr[OOqqg0gQqOGGO0G0oOGgGq00g00O0qQqo][oooOGQQ0QoqgOogGGO00O0o0OoGgGG00Q]){oG0Q0ggOQ
ooo0QqQQQo0o0g0o0gQgooQo
Ooooq0G0Q00qoqOq0GooGOqqOqOOQQqqo;}}}oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
qOoG0oq0OGggqGgOqGQQqqOO00OGOGQoo;}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isAntiTransitive(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgo
oQo (((*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg * *Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg) &
*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg) == empty());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isFull(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
((*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg | identity() |
!*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg) == full());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isAsymmetric(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
!(*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg ==
(!*Gg0gqO00GQogQ0oQQoQqoGO0QgGQ0QGqg));}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isTolerant(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
(isReflexive() && isSymmetric());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isEquivalent(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
(isReflexive() && isSymmetric() &&
isTransitive());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isOrder(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
(isAntiSymmetric() && isTransitive());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isWeakOrder(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
(isOrder() && isReflexive());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isStrictOrder(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
(isOrder() && isAntiReflexive());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
```

```
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isLinearOrder(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQgooQo
(isOrder() && isFull());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isWeakLinearOrder(){oG0Q0ggOQooo0QqQQQo0o0g0o0gQg
ooQo (isWeakOrder() && isFull());}oOoOGOgGogOoGGOggg0QGqogQ00GogqQO
oGgoqq0GoGGogQ0QqggO0qOoQOgqggQgg::isStrictLinearOrder(){oG0Q0ggOQooo0QqQQQo0o0g0o0g
QgooQo (isStrictOrder() && isFull());}
```

Вывод: в ходе расчетно-графического задания мы изучили способы обфускации кода, написали обфускатор строк кода на C++.