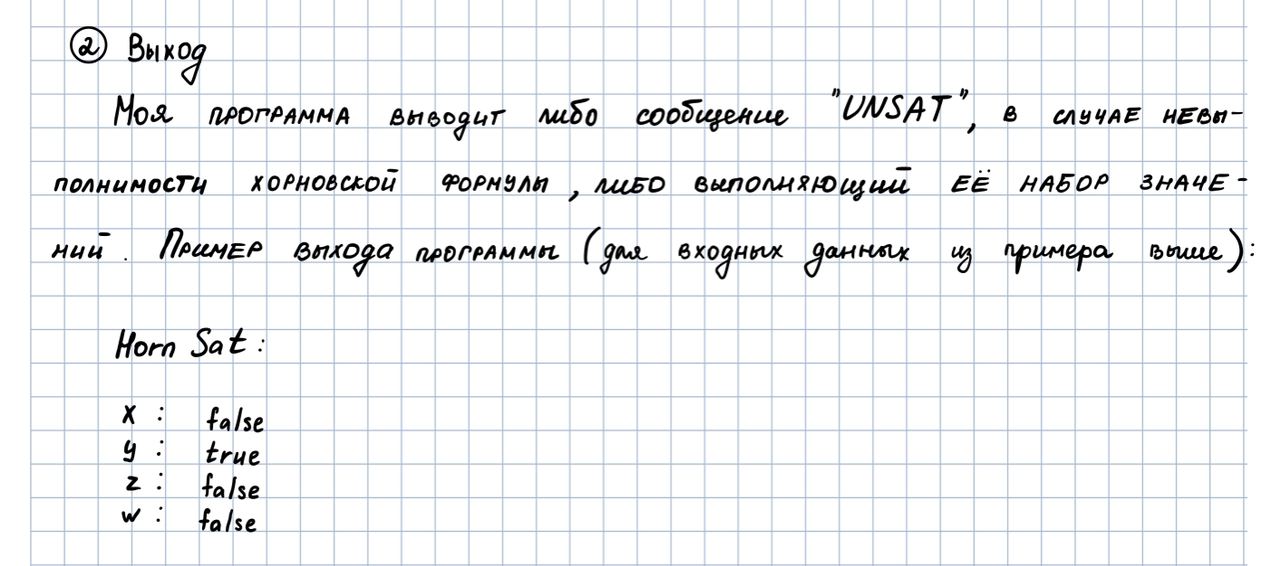
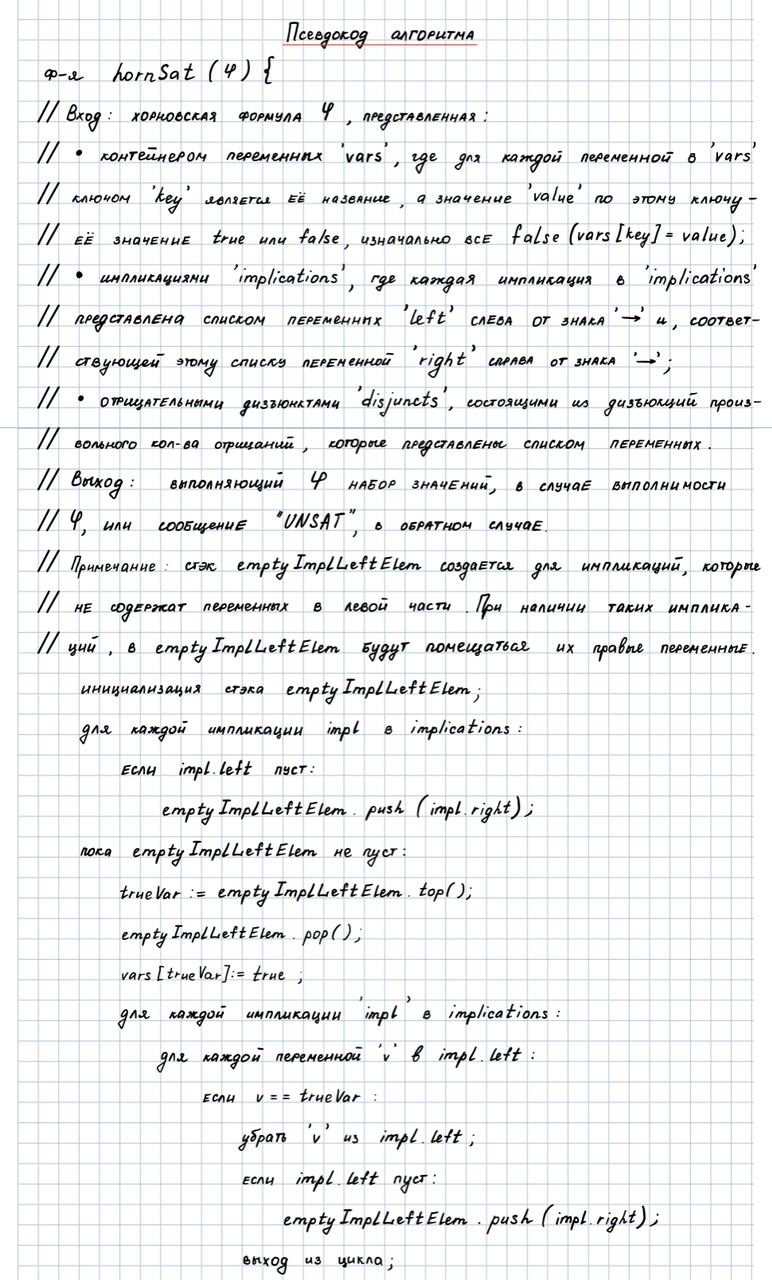
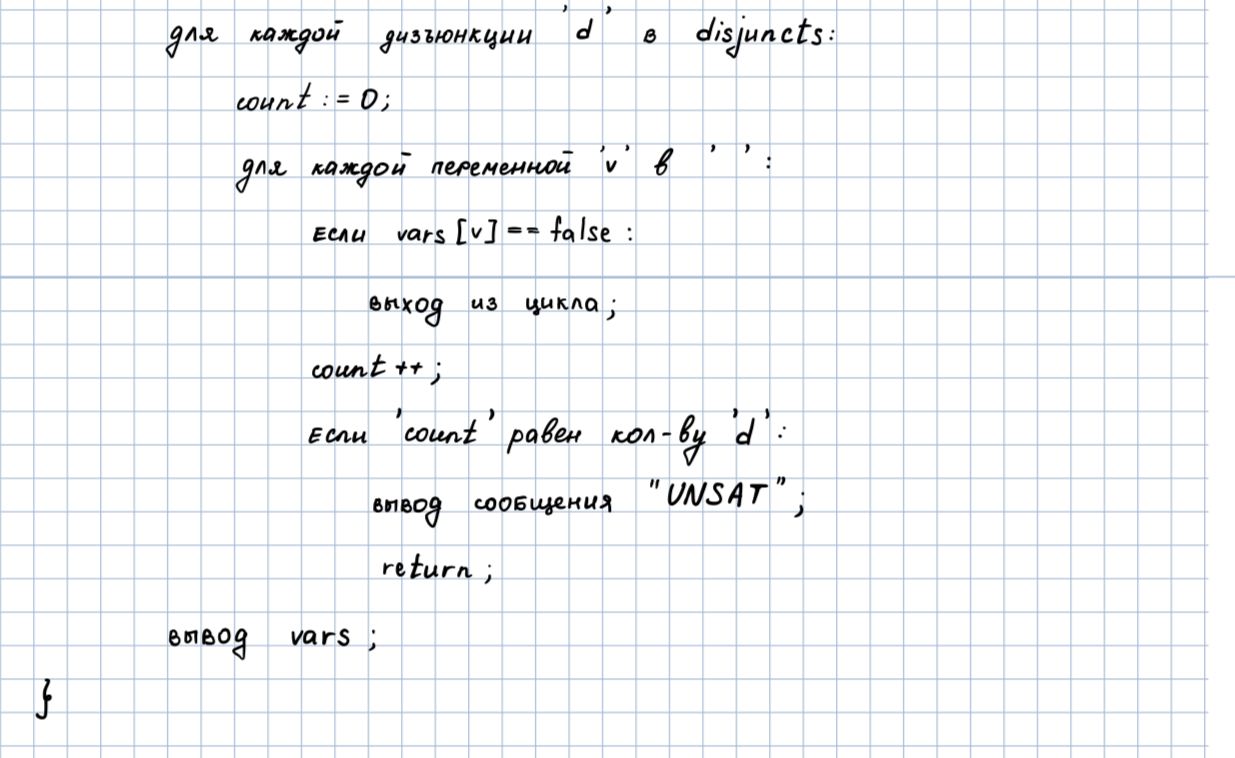
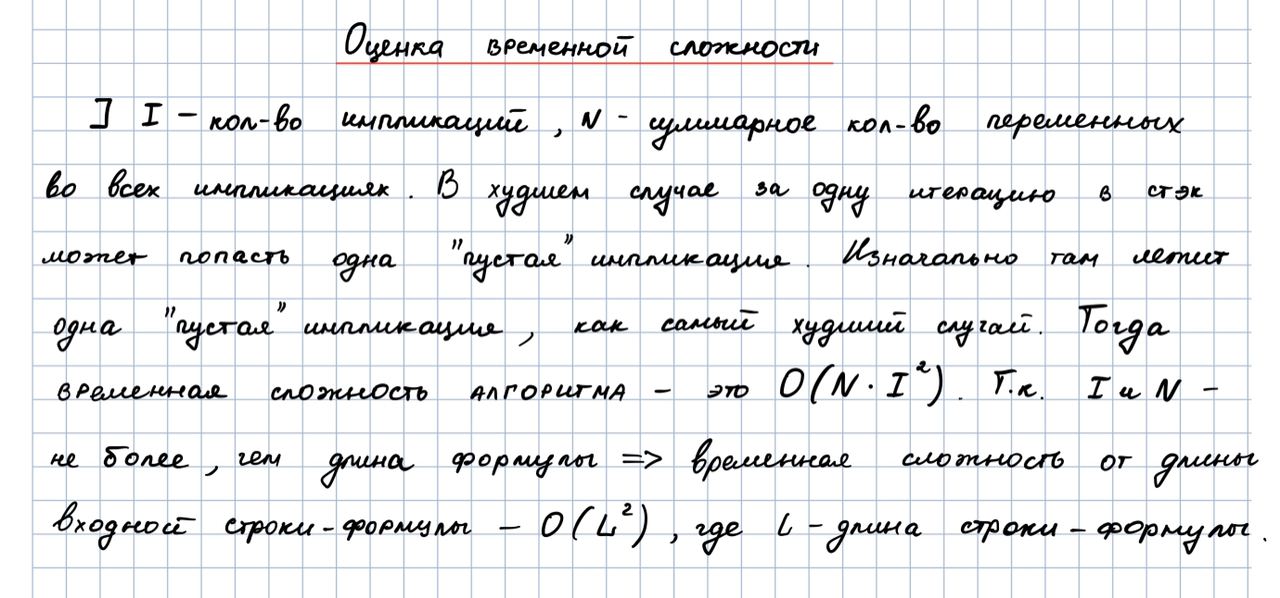
Задание 7











Код программы

#include <iostream>

#include <vector>

#include <stack>

#include <list>

#include <map>

**class** HornFormula

{

**public**:

HornFormula(**int** n, std::map<**char**, **bool**>& vars);

**void** addImplication(std::pair<std::list<**char**>, **char**> &impl);

**void** addDisjunction(std::list<**char**> &disjunction);

**void** printSolution();

**void** hornSat();

**private**:

**int** n\_;

std::map<**char**, **bool**> vars\_;

std::vector<std::pair<std::list<**char**>, **char**>> implications\_;

std::vector<std::list<**char**>> disjunctions\_;

};

HornFormula::HornFormula(**int** n, std::map<**char**, **bool**>& vars)

{

**this**->n\_ = n;

**this**->vars\_ = vars;

}

//Добавление импликации в хорновскую формулу

**void** HornFormula::addImplication(std::pair<std::list<**char**>, **char**> &implication)

{

implications\_.push\_back(implication);

}

//Добавление дизъюнкции в хорновскую формулу

**void** HornFormula::addDisjunction(std::list<**char**> &disjunction)

{

disjunctions\_.push\_back(disjunction);

}

//Фукнция проверяющая выполнимость хорновской формулы

**void** HornFormula::hornSat()

{

//стэк для элементов справа от '->' импликаций, в которых нет элементов

//слева от '->'

std::stack<**char**> emptyFirstElemImpl;

**for** (**auto** i: implications\_)

{

**if** (i.first.empty())

{

emptyFirstElemImpl.push(i.second);

}

}

**while** (!emptyFirstElemImpl.empty()) //пока есть не выполненная импликация

{

**char** trueVar = emptyFirstElemImpl.top();

emptyFirstElemImpl.pop();

vars\_[trueVar] = **true**; //присваивание переменной справа значения true

**for** (**auto**& i: implications\_)

{

**for** (**auto** j = i.first.begin(); j != i.first.end(); j++)

{

**if** (\*j == trueVar)

{

i.first.erase(j);

**if** (i.first.empty())

{

emptyFirstElemImpl.push(i.second);

}

**break**;

}

}

}

}

**for** (**auto** i: disjunctions\_)

{

**int** count = 0;

**for** (**auto** j: i)

{

**if** (vars\_[j] == **false**)

{

**break**;

}

count++; //количество положительных переменных дизъюнкции

**if** (count == i.size())

{

std::cout << "\nUNSAT\n";

**return**; //формула невыполнима

}

}

}

printSolution();

**return**;

}

//Функция вывода решения в консоль

**void** HornFormula::printSolution()

{

std::cout << "\nHornSat:\n";

**for** (**auto** i: vars\_)

{

std::cout << i.first << ": ";

**if** (i.second == **true**)

{

std::cout << "true" << '\n';

}

**else**

{

std::cout << "false" << '\n';

}

}

}

**int** main()

{

std::map<**char**, **bool**> elements;

**int** nElements = 0;

**char** elem;

std::cin >> nElements;

**for** (**int** i = 0; i < nElements; i++)

{

std::cin >> elem;

elements[elem] = **false**;

}

HornFormula hornFormula(nElements, elements);

**int** nImlications = 0;

std::cin >> nImlications;

**for** (**int** i = 0; i < nImlications; i++)

{

std::cin >> elem;

std::pair<std::list<**char**>, **char**> implication;

**while** (elem != '>')

{

implication.first.push\_back(elem);

std::cin >> elem;

}

std::cin >> elem;

**while** (elem != ';')

{

implication.second = elem;

std::cin >> elem;

}

hornFormula.addImplication(implication);

}

**int** nDisjunctions = 0;

std::cin >> nDisjunctions;

**for** (**int** i = 0; i < nDisjunctions; i++)

{

std::cin >> elem;

std::list<**char**> disjunction;

**while** (elem != ';')

{

disjunction.push\_back(elem);

std::cin >> elem;

}

hornFormula.addDisjunction(disjunction);

}

hornFormula.hornSat();

**return** 0;

}

Тесты

|  |  |
| --- | --- |
| Входные данные | Выходные данные  (с пояснением) |
| ϕ = (w∧y∧z→x) ∧ (x∧z→w) ∧ (x→y) ∧ (→x) ∧ (x∧y→w) ∧ (∨∨) ∧ | Начинаем со всех значений false и далее видим, что ‘x’ должно быть true в силу импликации ‘→x’. Импликация ‘x→y’ вынуждает присвоить ‘y’ значение true. Т.к. ‘x’ и ‘y’ – true, импликация ‘x∧y→w’ вынуждает присвоить ‘w’ значение true. Т.о. дизъюнкт ‘∨∨’ ложен ⇒ ϕ невыполнима. |
| 4  x y z w  5  w y z > x;  x z > w;  x > y;  >x;  x y > w;  2  w x y;  z; | UNSAT |
| ϕ = (w∧y∧z→x) ∧ (x∧z→w) ∧ (x→y) ∧ (→x) ∧ (x∧y→w) ∧ (∨∨∧) | Начинаем со всех значений false и далее видим, что ‘x’ должно быть true в силу импликации ‘→x’. Импликация ‘x→y’ вынуждает присвоить ‘y’ значение true. Т.к. ‘x’ и ‘y’ – true, импликация ‘x∧y→w’ вынуждает присвоить ‘w’ значение true. Т.о. импликация ‘x∧z→w’ – истинна, как и дизъюнкт ‘∨∨∧’ ⇒ ϕ выполнима. |
| 4  x y z w  5  w y z > x;  x z > w;  x > y;  >x;  x y > w;  1  w x y z; | HornSat:  w: true  x: true  y: true  z: false |
| ϕ = (x→y) ∧ (→y) ∧ (∨) | Начинаем со всех значений false и далее видим, что ‘y’ должно быть true в силу импликации ‘→y’. С выбранным значением для ‘y’ импликация ‘x→y’ – истинна, как и дизъюнкт ‘∨’ ⇒ ϕ выполнима. |
| 2  x y  2  x > y;  > y;  1  x y; | HornSat:  x: false  y: true |
| ϕ = (x→y) ∧ (y→x) ∧ | Начинаем со всех значений false и далее видим, что с выбранными значениями все условия формулы истины ⇒ ϕ выполнима. |
| 2  x y  2  x > y;  y > x;  1  x; | HornSat:  x: false  y: false |

Задание 8

