Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
"Белгородский государственный технологический университет им. В. Г. Шухова"
(БГТУ им. В.Г. Шухова)

Институт информационных технологий и управляющих систем

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 2
по дисциплине математическая логика и теория алгоритмов
тема: Логика предикатов

Выполнил: студент группы ПВ-223
Игнатьев Артур Олегович
Проверил: старший преподаватель
Куценко Дмитрий Александрович

Белгород 2023

Лабораторная работа № 2

Тема: Логика предикатов

Цель работы: Разработать программу, способную считывать несколько формул-посылок логики высказываний и выводить на экран все формулы-следствия из этих посылок.

Содержание отсчёта

1. Название и цель лабораторной работы.

2. Решение предложенных в теоретической части задач.

3. Программа на выбранном языке программирования в виде исходных кодов (с поясняющими комментариями) и в электронном варианте для демонстрации на ЭВМ.

4. Спецификация программы с указанием основных структур данных и алгоритмов.

5. Примеры работы программы на тестовых данных.

Вариант 11

Теоретическое задание:

2.4. Записать с помощью предиката равенства $E(x, y) -$ «x равен y», определённого на множестве натуральных чисел, используя функцию умножения $p(x, y) = x * y$: если один из двух сомножителей делится на некоторое число z, то на него делится и произведение.

8.3. Определить, выполнимы ли следующие формулы:
$\exists x \forall y \left(Q(x, x) \& \overline{Q(x, y)}\right)$

12.5. Определить, какие из следующих формул тождественно истинны:
$\forall x \left(Q(x) \to P(x)\right) \leftrightarrow \left(\exists x Q(x) \to \forall x P(x)\right)$

24.11. Привести к предварённой нормальной форме: $\forall x (A(x) \to \exists y B(y))$

41. Идёт дождь или жарко. Если дождь идёт, то жарко. Если не идёт дождь, то не жарко. Верно ли, что если жарко, то не должен идти дождь? Проверить это с помощью метода резолюций.

Практическое задание:

Вариант 1. Разработать программу, способную считывать несколько формул-посылок логики высказываний и выводить на экран все формулы-следствия из этих посылок.

Решение заданий:

Практическая часть:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <windows.h>

#define N 100
long unsigned d[N];
long unsigned c[N];
//выделение памяти матрице
int **getMemoryMatrix(int str, int tab) {
    int **matr = (int **)malloc(str * sizeof(int *));
    for (int i = 0; i < str; i++)
        matr[i] = (int *)malloc(tab * sizeof(int));
    return matr;
}
//ввод КНФ пользователем в заданном программой виде
void inputDnf(int** a, int* b, int m, int n) {
    int i, j;

    printf("Ваши перименные: ");
    for (i = 0; i<n; i++) {
        b[i] = 'A' + i;
        printf("%c ", b[i]);
    }
    printf("\n");
    printf("Обозначения:\n");
    printf("Наличие переменной : 1\n");
    printf("Отрицание переменной: -1\n");
    printf("Отсутствие переменной: 0 \n");

    for (i = 0; i < m; i++) {
        printf("Скобка %d:\n", i + 1);
        for (j = 0; j < n; j++)
            scanf("%i", &a[i][j]);
    }
}

//вывод ДНФ на экран
void outputForml(int **a, int* b, int m, int n) {
    int i = 0, j = 0;

    for (i = 0; i < m; i++) {
        printf("(");
        j = 0;
        while ((a[i][j] == 0) && (j < n))
            j++;
        if (j < n) {
            if (a[i][j] == -1)
```

```
                printf("!%c", b[j]);
            else
                printf("%c", b[j]);
        }
        j++;
        for (; j < n; j++) {
            if (a[i][j] == -1)
                printf(" + !%c", b[j]);
            if (a[i][j] == 1)
                printf(" + %c", b[j]);

        }
        printf(")");
        if ((i + 1) < m)
            printf("*");
    }
    printf("\n");
}

//вывод полученной таблицы истинности, а для данной ДНФ
int** outputTable(int **a, int *b, int m, int n) {
    int i, j, mask = 1, f, x, z, k;
    //вывод обозначение столбца в таблице истинности
    for (i = 0; i < n; i++)
        printf("%c ", b[i]);
    printf("f\n");

    int all = pow(2, n);
    //таблица истинности
    int **table = getMemoryMatrix(all, n + 1);


    for (i = 0; i < all; i++) {
        //получение двоичного вектора основываясь на предыдущем векторе
        for (j = 0; j < n; j++) {
            table[i][j] = (mask & (i >> (n - 1 - j)));
            printf("%i ", table[i][j]);
        }
        f = 1;
        z = 0;
        //высчитывание формулы по полученному двоичному вектору
        while ((z < m) && (f)) {
            x = 0;
            k = 0;
            while (k < n) {
                if (a[z][k] == 1)
                    x |= table[i][k];
                if (a[z][k] == -1)
                    x |= !table[i][k];
                k++;
            }
            f = f && x;
            z++;
        }
        table[i][n] = f;
        printf("%i\n", table[i][n]);
    }
    return table;
}


int outputSknf(int **table, int n, int* ABC, int *b) {
    int i, j, k = 1;
    int n1 = pow(2, n);
```

```
    for (i = 0; i < n1; i++)
        if (table[i][n] == 0) {
            b[k - 1] = i;
            printf("%i)", k);
            printf("(");
            for (j = 0; j<n - 1; j++) {
                if (table[i][j] == 1)
                    printf("!%c+", ABC[j]);
                else
                    printf("%c+", ABC[j]);
            }
            if (table[i][n - 1] == 1)
                printf("!%c", ABC[n - 1]);
            else
                printf("%c", ABC[n - 1]);

            printf(")\n");
            k++;
        }
    return k - 1;
}


void input(long unsigned c[], int k) {
    for (int i = 0; i < k; i++)
        c[i] = i + 1;
}


void output(long unsigned d[], int k, int* b, int** table, int n, int *ABC) {
    size_t i, j;
    for (i = 0; i < k; i++)
        if (d[i]) {
            printf("(");
            for (j = 0; j < n - 1; j++) {
                if (table[b[i]][j] == 1)
                    printf("!%c+", ABC[j]);
                else
                    printf("%c+", ABC[j]);
            }
            if (table[b[i]][n - 1] == 1)
                printf("!%c", ABC[n - 1]);
            else
                printf("%c", ABC[n - 1]);
            printf(")");
        }
}


void recurs(size_t i, int k, int* b, int** table, int n, int* ABC) {
    short unsigned x;
    for (x = 0; x <= 1; x++) {
        d[i] = x;
        if (i == k - 1) {
            output(d, k, b, table, n, ABC);
            printf("\n");
        }
        else
            recurs(i + 1, k, b, table, n, ABC);
    }
}
//выделение памяти массиву
int *getMemoryArray(int size) {
    return (int *)malloc(size * sizeof(int));
```

```
}
//очищение памяти массива
void freeMemoryArray(int *arr) {
    free(arr);
}


//очищение памяти матрицы
void freeMemoryMatrix(int **matrix, int str) {
    for (int i = 0; i < str; i++)
        free(matrix[i]);
    free(matrix);
}



int main() {
    SetConsoleOutputCP(CP_UTF8);

    int i, k;
    int **table;
    printf("Количество скобок в КНФ: ");
    int m;
    scanf("%i", &m);
    printf("Количество переменных: ");
    int n;
    scanf("%i", &n);
    int *b = getMemoryArray(pow(2, n));
    //матрица формулы ДНФ
    //строка матрицы - одна скобка ДНФ
    //каждый столбец логический связан с одной переменной
    //значение в ячейке характеризует наличие(1), отрицание(-1) или
отсутствие(0) переменной
    int **a = getMemoryMatrix(m, n);

    //массив переменных в виде символов
    int *ABC = getMemoryArray(n);

    //ввод ДНФ
    inputDnf(a, ABC, m, n);
    //вывод ДНФ в привычной форме
    outputForml(a, ABC, m, n);
    //построение таблицы истинности и вывод её на экран
    table = outputTable(a, ABC, m, n);
    k = outputSknf(table, n, ABC, b);
    input(c, k);
    recurs(0, k, b, table, n, ABC);

    freeMemoryArray(ABC);
    freeMemoryMatrix(a, m);
    freeMemoryMatrix(table, pow(2, m));
    return 0;
}
```

Вывод: на этой лабораторной работе я разработал программу, способную считывать несколько формул-посылок логики высказываний и выводить на экран все формулы-следствия из этих посылок.