

Домашнє завдання №4

Виконати реалізацію моделі машини Тюрінга на C/C++ для виконання обчислень згідно власного варіанту.

Вибір варіанту

$$(N_{\text{ж}} + N_{\text{г}} + 1) \% 7 + 1$$

де: $N_{\text{ж}}$ – порядковий номер студента в групі, а $N_{\text{г}}$ – номер групи (1,2,3,4,5,6,7 або 8)

Варіанти завдання

№	Формат лексем
1	X NAND Y
2	X OR Y
3	X NOR Y
4	NOT X
5	NOT Y
6	0
7	1

Приклад коду

Наведений зразок коду реалізовує операцію кон'юнкції ($X \wedge Y$). Потрібно модифікувати наведений зразок коду для реалізації завдання згідно власного варіанту.

Лістинг

```
#include <stdio.h>

#define DECLSTATE(NAME, ...) typedef enum {__VA_ARGS__, size##NAME} NAME;

#define GET_ENUM_SIZE(NAME) size##NAME

DECLSTATE(A,
a0,
v0/* 0 */,
v1/* 1 */,
sT/* ^ */
)

DECLSTATE(Q,
q0,
q1,
q2,
q3,
q4,
q5,
q6,
q7,
q8,
qf
```

```

)

#define NO_ACTION 0x7f

#define NO_RULE {NO_ACTION, NO_ACTION, NO_ACTION}

#define S 0
#define L 1
#define R 2

typedef unsigned char INSTRUCTION[3];
typedef INSTRUCTION PROGRAM[GET_ENUM_SIZE(A)][GET_ENUM_SIZE(Q)];

PROGRAM initProgram = { /* default pass */
    //          q0          q1          q2
    q3          q4          q5          q6
    q7          q8
    /* a0 */{ { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, {
NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, {
NO_ACTION, R, q1 }, { NO_ACTION, R, q1 } },
    /* 0 */{ { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, {
NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, {
NO_ACTION, R, q1 }, { NO_ACTION, R, q1 } },
    /* 1 */{ { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, {
NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, { NO_ACTION, R, q1 }, {
NO_ACTION, R, q1 }, { NO_ACTION, R, q1 } },
    /* ^ */{ { NO_ACTION, S, q0 }, { NO_ACTION, S, q0 }, { NO_ACTION, S, q0 }, {
NO_ACTION, S, q0 }, { NO_ACTION, S, q0 }, { NO_ACTION, S, q0 }, { NO_ACTION, S, q0 }, {
NO_ACTION, S, q0 }, { NO_ACTION, S, q0 } }
};

PROGRAM program = { /* default pass */
    //          q0          q1          q2
    q3          q4          q5          q6
    q7          q8
    /* a0 */{ { NO_ACTION, L, q1 }, NO_RULE, NO_RULE,
NO_RULE, { NO_ACTION, L, q4 }, { NO_ACTION, L, q5 }, { v1, R, q8 }, {
v0, R, q8 }, { NO_ACTION, R, q8 } },
    /* 0 */{ { NO_ACTION, R, q0 }, { a0, L, q3 }, { NO_ACTION, L, q2 }, {
NO_ACTION, L, q3 }, { a0, L, q7 }, { a0, L, q7 }, { NO_ACTION, L, q6 }, {
NO_ACTION, L, q7 }, { NO_ACTION, R, q8 } },
    /* 1 */{ { NO_ACTION, R, q0 }, { a0, L, q2 }, { NO_ACTION, L, q2 }, {
NO_ACTION, L, q3 }, { a0, L, q6 }, { a0, L, q7 }, { NO_ACTION, L, q6 }, {
NO_ACTION, L, q7 }, { NO_ACTION, R, q8 } },
    /* ^ */{ { NO_ACTION, R, q0 }, { NO_ACTION, S, qf }, { NO_ACTION, L, q4 }, {
NO_ACTION, L, q5 }, NO_RULE, NO_RULE, NO_RULE,
NO_RULE, { NO_ACTION, R, q0 } }
};

typedef unsigned char tapeElementType;
#define TAPE_SIZE 256

// #define MT_BEGIN_POSITION_STATE 127

typedef struct structMT{
    tapeElementType tape[TAPE_SIZE];
    PROGRAM * initProgram;
    PROGRAM * program;
    void(*stepRun)(struct structMT * mt, PROGRAM program);
    void(*run)(struct structMT * mt);
    void(*statePrint)(struct structMT * mt);
    unsigned char * debugType;
    Q state;
    unsigned int positionState;

```

```

} MT;

void stepRunner(MT * mt, PROGRAM program){
    INSTRUCTION * instruction;

    if (!mt){
        return;
    }

    instruction = program[mt->tape[mt->positionState]][mt->state];

    if ((*instruction)[0] != NO_ACTION){
        mt->tape[mt->positionState] = (*instruction)[0];
    }

    if ((*instruction)[1] == L){
        --mt->positionState;
    }
    else if ((*instruction)[1] == R){
        ++mt->positionState;
    }

    mt->state = (*instruction)[2];
}

void runner(MT * mt){
    if (!mt){
        return;
    }

    // init
    mt->state = q0;
    mt->positionState = 0;
    do{
        mt->stepRun(mt, mt->initProgram);
    } while (mt->state != q0);

    printf("Begin tape state:\r\n");
    mt->statePrint(mt);
    printf("\r\n\r\n");

    printf("Tape state:\r\n");
    // run program
    while (mt->state != qf){
        mt->stepRun(mt, mt->program);
        if (mt->statePrint){
            mt->statePrint(mt);
            if (*mt->debugType == 2){
                _sleep(250);
            }
            else if (*mt->debugType == 3){
                getchar();
            }
            else{
                _sleep(25);
            }
        }
    }
}

#define MAX_PRINT_TAPE_ELEMENT_COUT 14
void statePrinter(MT * mt){
    unsigned int index = 0;

```

```

    if (!mt){
        return;
    }

    for (; index < MAX_PRINT_TAPE_ELEMENT_COUNT; ++index){
        switch (mt->tape[index]){
            case a0:
                if (index == mt->positionState)
                    printf("[a0] ");
                else{
                    printf(" a0 ");
                }
                break;
            case v0:
                if (index == mt->positionState)
                    printf("[0] ");
                else{
                    printf(" 0 ");
                }
                break;
            case v1:
                if (index == mt->positionState)
                    printf("[1] ");
                else{
                    printf(" 1 ");
                }
                break;
            case sT:
                if (index == mt->positionState)
                    printf("[^] ");
                else{
                    printf(" ^ ");
                }
                break;
            default:
                if (index == mt->positionState)
                    printf("[?] ");
                else{
                    printf(" ? ");
                }
                break;
        }
    }

    if (mt->state == qf){
        printf("    qf");
    }
    else{
        printf("    q%d", mt->state);
    }

    printf("\n");
}

tapeElementType tape[TAPE_SIZE];

#define INIT_TAPE_DATE { a0, a0, a0, a0, v1, v0, v1, v1, sT, v1, v0, v1, v1 }

int main(){
    int ch;
    MT mt = { INIT_TAPE_DATE, initProgram, program, stepRunner, runner, statePrinter,
(char*)&ch };

```

```
    printf("mode: \r\ndefault - run all\r\n    '2' - live\r\n    '3' - live by press  
    \nEnter\\r\n");  
    ch = getchar();  
    ch -= '0';  
  
    mt.run(&mt);  
    getchar();  
  
    return 0;  
}
```