```cpp
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ctype.h>

#define MAX_STRUCTS 30
#define HASH_LENGTH 10

//CAIO LIMA E SOUZA DELLA TORRE SANCHES - 17225285

union var_valor {
    int i;
    float f;
    char c;
};

typedef struct {
    int tipo;
    char nome[30];
    var_valor valor;
    //int valor;
} var;

typedef struct {
    var info;
    int link;
} ll;

enum error_type {undeclared_var = 1, malformed_expression, unknown_type, unknown_op};
void throw_err(error_type e, int line);

int exec(ll *mem, int *pri, int *disp, const char *exp, int line);
int exec(ll *mem, int *pri, int *disp, char *exp, int line);

int insert(var d, ll *mem, int *pri, int *disp);
int hash(char nome[]);
void reset_structs(ll *mem, int *pri, int *disp);
int busca(ll *mem, int *pri, char nome_busca[]);
void exibe(ll *mem, int *pri, int disp);

int main() {
    FILE *fd = fopen("prog.txt", "r");
    int pri[HASH_LENGTH];
    ll mem[MAX_STRUCTS];
    int disp;
    int line = 1;
    char instr[30];
    char last_char;
    reset_structs(mem, pri, &disp);

    int exec_ok = 1;
    printf("-----START OF PROGRAM-----\n");
    while (exec_ok && !feof(fd)) {
        fgets(instr, 30, fd);
        printf("%s", instr);
        exec_ok = exec(mem, pri, &disp, instr, line);
        last_char = fgetc(fd);
        if (!feof(fd)) ungetc(last_char, fd);
        printf("\n");
        line++;
    }
    printf("-----END OF PROGRAM-----\n");
}

```

```cpp
void throw_err(error_type e, int line) {
    switch (e) {
        case undeclared_var:
            printf("Undeclared variable @ %d\n", line); break;
        case malformed_expression:
            printf("Malformed expression @ %d\n", line); break;
        case unknown_type:
            printf("Unknown variable type @ %d\n", line); break;
        case unknown_op:
            printf("Unknown operation @ %d\n", line); break;
        default:
            printf("Uknown error @ %d\n", line); break;
    }
}

int exec(ll *mem, int *pri, int *disp, const char *exp, int line) {
    char temp[30];
    strcpy(temp, exp);
    exec(mem, pri, disp, temp, line);
}

int exec(ll *mem, int *pri, int *disp, char *exp, int line) {
    char *t = exp;
    int i;
    char res[30], op1[30], op2[30], op;
    int pres, pop1, pop2;
    i = 0;
    while (isalpha(*t)) { res[i++] = *t; t++; }
    res[i] = '\0';
    if (!strcmp(res, "read")) {
        if (*t != '(') { throw_err(malformed_expression, line); return 0; }
        t++;

        do {
            i = 0;
            while (isalpha(*t)) { op1[i++] = *t; t++; }
            op1[i] = '\0';

            i = busca(mem, pri, op1);
            if (i == -1) { throw_err(undeclared_var, line); return 0; }
            switch (mem[i].info.tipo) {
                case 0: printf("int   %5s = ", op1); scanf("%d", &(mem[i].info.valor.i)); break;
                case 1: printf("float %5s = ", op1); scanf("%f", &(mem[i].info.valor.f)); break;
                case 2: printf("char  %5s = ", op1); scanf("%c", &(mem[i].info.valor.c)); break;
                //case 1: printf("float %5s = ", op1);
                //  float temp_f;
                //  scanf("%f", &temp_f);
                //  mem[i].info.valor = (int)temp_f;
                //  break;
                //case 2: printf("char  %5s = ", op1);
                //  char temp_c;
                //  scanf("%c", &temp_c);
                //  mem[i].info.valor = (int)temp_c;
                //  break;
                default: throw_err(unknown_type, line); return 0;
            }

            if (*t != ';') {
                if (*t != ',' && *t != ')') { throw_err(malformed_expression, line); return 0; }
                t++;
            }
        } while (*t != ';');
        //printf("\n");

    } else if (!strcmp(res, "print")) {
        if (*t != '(') { throw_err(malformed_expression, line); return 0; }
        t++;

        do {
            i = 0;
            while (isalpha(*t)) { op1[i++] = *t; t++; }
            op1[i] = '\0';

            i = busca(mem, pri, op1);
            if (i == -1) { throw_err(undeclared_var, line); return 0; }
            switch (mem[i].info.tipo) {
                case 0: printf("int   %5s = %d\n", op1, mem[i].info.valor.i); break;
                case 1: printf("float %5s = %f\n", op1, mem[i].info.valor.f); break;
                case 2: printf("char  %5s = %c\n", op1, mem[i].info.valor.c); break;
                default: throw_err(unknown_type, line); return 0;
            }

            if (*t != ';') {
                if (*t != ',' && *t != ')') { throw_err(malformed_expression, line); return 0; }
                t++;
            }
        } while (*t != ';');
        //printf("\n");

    } else {
        if (!strcmp(res, "int")) {
            if (*t != ' ') { throw_err(malformed_expression, line); return 0; }
            t++;
            i = 0;
            var temp_var;
            do {
                temp_var.tipo = 0;
                temp_var.valor.i = 0;

                i = 0;
                while (isalpha(*t)) { op1[i++] = *t; t++; }
                op1[i] = '\0';

                strcpy(temp_var.nome, op1);
                insert(temp_var, mem, pri, disp);
                if (*t != ';') {
                    if (*t != ',') { throw_err(malformed_expression, line); return 0; }
                    t++;
                }
            } while (*t != ';');

        } else if (!strcmp(res, "float")) {
            if (*t != ' ') { throw_err(malformed_expression, line); return 0; }
            t++;
            i = 0;
            var temp_var;
            do {
                temp_var.tipo = 1;
                temp_var.valor.i = 0;

                i = 0;
                while (isalpha(*t)) { op1[i++] = *t; t++; }
                op1[i] = '\0';

                strcpy(temp_var.nome, op1);
                insert(temp_var, mem, pri, disp);
                if (*t != ';') {
                    if (*t != ',') { throw_err(malformed_expression, line); return 0; }
                    t++;
                }
            } while (*t != ';');

        } else if (!strcmp(res, "char")) {
            if (*t != ' ') { throw_err(malformed_expression, line); return 0; }
            t++;
            i = 0;
            var temp_var;
            do {
                temp_var.tipo = 2;
                temp_var.valor.i = 0;

                i = 0;
                while (isalpha(*t)) { op1[i++] = *t; t++; }
                op1[i] = '\0';

                strcpy(temp_var.nome, op1);
                insert(temp_var, mem, pri, disp);
                if (*t != ';') {
                    if (*t != ',') { throw_err(malformed_expression, line); return 0; }
                    t++;
                }
            } while (*t != ';');

        } else {
            pres = busca(mem, pri, res);
            if (*t != '=') { throw_err(malformed_expression, line); return 0; }
            t++;

            i = 0;
            while (isalpha(*t)) { op1[i++] = *t; t++; }
            op1[i] = '\0';
            pop1 = busca(mem, pri, op1);
            op = *t;
            t++;

            i = 0;
            while (isalpha(*t)) { op2[i++] = *t; t++; }
            op2[i] = '\0';
            pop2 = busca(mem, pri, op2);
            if (*t != ';') { throw_err(malformed_expression, line); return 0; }

            switch (op) {
                case '+':
                    mem[pres].info.valor.i = mem[pop1].info.valor.i + mem[pop2].info.valor.i;
                    break;
                case '-':
                    mem[pres].info.valor.i = mem[pop1].info.valor.i - mem[pop2].info.valor.i;
                    break;
                case '*':
                    mem[pres].info.valor.i = mem[pop1].info.valor.i * mem
```

```cpp
[pop2].info.valor.i;
                    break;
                case '/':
                    mem[pres].info.valor.i = mem[pop1].info.valor.i / mem[pop2].info.valor.i;
                    break;
                default:
                    throw_err(unknown_op, line); return 0;
            }
        }
    }
    return 1;
}

int insert(var d, ll *mem, int *pri, int *disp) {
    int novo, p;
    if (*disp == -1) return 0; //Nao cabe
    novo = *disp; *disp = mem[*disp].link;
    mem[novo].info = d;
    p = hash(d.nome);
    mem[novo].link = pri[p];
    pri[p] = novo;
    return 1;
}

int hash(char nome[]) {
    int i = 0;
    int soma = 0;
    while (nome[i] != '\0') soma += (int)nome[i++];
    return soma % HASH_LENGTH;
}

void reset_structs(ll *mem, int *pri, int *disp) {
    int i;
    var zero;
    zero.tipo = 0;
    zero.valor.i = 0;
    strcpy(zero.nome, "");
    for (i = 0; i < HASH_LENGTH; i++) pri[i] = -1;
    for (i = 0; i < MAX_STRUCTS - 1; i++) { mem[i].link = i + 1; mem[i].info = zero; }
    mem[i].link = -1;
    mem[i].info = zero;
    *disp = 0;
}

int busca(ll *mem, int *pri, char nome_busca[]) {
    int p, x;
    p = hash(nome_busca);
    x = pri[p];
    while (x != -1) {
        if (!strcmp(mem[x].info.nome, nome_busca)) return x;
        x = mem[x].link;
    }
    return -1;
}

void exibe(ll *mem, int *pri, int disp) {
    int h;
    for (h = 0; h < HASH_LENGTH; h++) {
        printf("\t%2d - [%2d]\n", h, pri[h]);
    }
    printf("\n");
    for (h = 0; h < MAX_STRUCTS; h++) {
        printf("\t%2d - %c[%2d | %10s = %3d][%2d]\n", h, (h == disp ? '>' : ' '), mem[h].info.tipo, mem[h].info.nome, mem[h].info.valor.i, mem[h].link);
    }
    printf("\n");
}
```