```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4
5  #include <ctype.h>
6
7  #define MAX_STRUCTS 30
8  #define HASH_LENGTH 10
9
10 //CAIO LIMA E SOUZA DELLA TORRE SANCHES - 17225285
11
12 union var_valor {
13     int i;
14     float f;
15     char c;
16 };
17
18 typedef struct {
19     int tipo;
20     char nome[30];
21     var_valor valor;
22     //int valor;
23 } var;
24
25 typedef struct {
26     var info;
27     int link;
28 } ll;
29
30 enum error_type {undeclared_var = 1, malformed_expression, unknown_type,
   unknown_op};
31 void throw_err(error_type e, int line);
32
33 int exec(ll *mem, int *pri, int *disp, const char *exp, int line);
34 int exec(ll *mem, int *pri, int *disp, char *exp, int line);
35
36 int insert(var d, ll *mem, int *pri, int *disp);
37 int hash(char nome[]);
38 void reset_structs(ll *mem, int *pri, int *disp);
39 int busca(ll *mem, int *pri, char nome_busca[]);
40 void exibe(ll *mem, int *pri, int disp);
41
42 int main() {
43     FILE *fd = fopen("prog.txt", "r");
44     int pri[HASH_LENGTH];
45     ll mem[MAX_STRUCTS];
46     int disp;
47     int line = 1;
48     char instr[30];
49     char last_char;
50     reset_structs(mem, pri, &disp);
51
52     int exec_ok = 1;
53     printf("-----START OF PROGRAM-----\n");
54     while (exec_ok && !feof(fd)) {
55         fgets(instr, 30, fd);
56         printf("%s", instr);
57         exec_ok = exec(mem, pri, &disp, instr, line);
58         last_char = fgetc(fd);
59         if (!feof(fd)) ungetc(last_char, fd);
60         printf("\n");
61         line++;
62     }
63     printf("-----END OF PROGRAM-----\n");
64 }
65
```

```
66  void throw_err(error_type e, int line) {
67      switch (e) {
68          case undeclared_var:
69              printf("Undeclared variable @ %d\n", line); break;
70          case malformed_expression:
71              printf("Malformed expression @ %d\n", line); break;
72          case unknown_type:
73              printf("Unknown variable type @ %d\n", line); break;
74          case unknown_op:
75              printf("Unknown operation @ %d\n", line); break;
76          default:
77              printf("Uknown error @ %d\n", line); break;
78      }
79  }
80
81  int exec(ll *mem, int *pri, int *disp, const char *exp, int line) {
82      char temp[30];
83      strcpy(temp, exp);
84      exec(mem, pri, disp, temp, line);
85  }
86
87  int exec(ll *mem, int *pri, int *disp, char *exp, int line) {
88      char *t = exp;
89      int i;
90      char res[30], op1[30], op2[30], op;
91      int pres, pop1, pop2;
92      i = 0;
93      while (isalpha(*t)) { res[i++] = *t; t++; }
94      res[i] = '\0';
95      if (!strcmp(res, "read")) {
96          if (*t != '(') { throw_err(malformed_expression, line); return 0; }
97          t++;
98
99          do {
100             i = 0;
101             while (isalpha(*t)) { op1[i++] = *t; t++; }
102             op1[i] = '\0';
103
104             i = busca(mem, pri, op1);
105             if (i == -1) { throw_err(undeclared_var, line); return 0; }
106             switch (mem[i].info.tipo) {
107                 case 0: printf("int   %5s = ", op1); scanf("%d", &(mem
    [i].info.valor.i)); break;
108                 case 1: printf("float %5s = ", op1); scanf("%f", &(mem
    [i].info.valor.f)); break;
109                 case 2: printf("char  %5s = ", op1); scanf("%c", &(mem
    [i].info.valor.c)); break;
110                 //case 1: printf("float %5s = ", op1);
111                 //  float temp_f;
112                 //  scanf("%f", &temp_f);
113                 //  mem[i].info.valor = (int)temp_f;
114                 //  break;
115                 //case 2: printf("char  %5s = ", op1);
116                 //  char temp_c;
117                 //  scanf("%c", &temp_c);
118                 //  mem[i].info.valor = (int)temp_c;
119                 //  break;
120                 default: throw_err(unknown_type, line); return 0;
121             }
122
123             if (*t != ';') {
124                 if (*t != ',' && *t != ')') { throw_err(malformed_expression,
    line); return 0; }
125                 t++;
126             }
127         } while (*t != ';');
```

```
128         //printf("\n");
129
130     } else if (!strcmp(res, "print")) {
131         if (*t != '(') { throw_err(malformed_expression, line); return 0; }
132         t++;
133
134         do {
135             i = 0;
136             while (isalpha(*t)) { op1[i++] = *t; t++; }
137             op1[i] = '\0';
138
139             i = busca(mem, pri, op1);
140             if (i == -1) { throw_err(undeclared_var, line); return 0; }
141             switch (mem[i].info.tipo) {
142                 case 0: printf("int   %5s = %d\n", op1, mem[i].info.valor.i);
    break;
143                 case 1: printf("float %5s = %f\n", op1, mem[i].info.valor.f);
    break;
144                 case 2: printf("char  %5s = %c\n", op1, mem[i].info.valor.c);
    break;
145                 default: throw_err(unknown_type, line); return 0;
146             }
147
148             if (*t != ';') {
149                 if (*t != ',' && *t != ')') { throw_err(malformed_expression,
    line); return 0; }
150                 t++;
151             }
152         } while (*t != ';');
153         //printf("\n");
154
155     } else {
156         if (!strcmp(res, "int")) {
157             if (*t != ' ') { throw_err(malformed_expression, line); return
    0; }
158             t++;
159             i = 0;
160             var temp_var;
161             do {
162                 temp_var.tipo = 0;
163                 temp_var.valor.i = 0;
164
165                 i = 0;
166                 while (isalpha(*t)) { op1[i++] = *t; t++; }
167                 op1[i] = '\0';
168
169                 strcpy(temp_var.nome, op1);
170                 insert(temp_var, mem, pri, disp);
171                 if (*t != ';') {
172                     if (*t != ',') { throw_err(malformed_expression, line);
    return 0; }
173                     t++;
174                 }
175             } while (*t != ';');
176
177         } else if (!strcmp(res, "float")) {
178             if (*t != ' ') { throw_err(malformed_expression, line); return
    0; }
179             t++;
180             i = 0;
181             var temp_var;
182             do {
183                 temp_var.tipo = 1;
184                 temp_var.valor.i = 0;
185
186                 i = 0;
```

```
187                 while (isalpha(*t)) { op1[i++] = *t; t++; }
188                 op1[i] = '\0';
189 
190                 strcpy(temp_var.nome, op1);
191                 insert(temp_var, mem, pri, disp);
192                 if (*t != ';') {
193                     if (*t != ',') { throw_err(malformed_expression, line);
    return 0; }
194                     t++;
195                 }
196             } while (*t != ';');
197 
198         } else if (!strcmp(res, "char")) {
199             if (*t != ' ') { throw_err(malformed_expression, line); return
    0; }
200             t++;
201             i = 0;
202             var temp_var;
203             do {
204                 temp_var.tipo = 2;
205                 temp_var.valor.i = 0;
206 
207                 i = 0;
208                 while (isalpha(*t)) { op1[i++] = *t; t++; }
209                 op1[i] = '\0';
210 
211                 strcpy(temp_var.nome, op1);
212                 insert(temp_var, mem, pri, disp);
213                 if (*t != ';') {
214                     if (*t != ',') { throw_err(malformed_expression, line);
    return 0; }
215                     t++;
216                 }
217             } while (*t != ';');
218 
219         } else {
220             pres = busca(mem, pri, res);
221             if (*t != '=') { throw_err(malformed_expression, line); return
    0; }
222             t++;
223 
224             i = 0;
225             while (isalpha(*t)) { op1[i++] = *t; t++; }
226             op1[i] = '\0';
227             pop1 = busca(mem, pri, op1);
228             op = *t;
229             t++;
230 
231             i = 0;
232             while (isalpha(*t)) { op2[i++] = *t; t++; }
233             op2[i] = '\0';
234             pop2 = busca(mem, pri, op2);
235             if (*t != ';') { throw_err(malformed_expression, line); return
    0; }
236 
237             switch (op) {
238                 case '+':
239                     mem[pres].info.valor.i = mem[pop1].info.valor.i + mem
    [pop2].info.valor.i;
240                     break;
241                 case '-':
242                     mem[pres].info.valor.i = mem[pop1].info.valor.i - mem
    [pop2].info.valor.i;
243                     break;
244                 case '*':
245                     mem[pres].info.valor.i = mem[pop1].info.valor.i * mem
```

```
    [pop2].info.valor.i;
246                     break;
247                 case '/':
248                     mem[pres].info.valor.i = mem[pop1].info.valor.i / mem
    [pop2].info.valor.i;
249                     break;
250                 default:
251                     throw_err(unknown_op, line); return 0;
252             }
253         }
254     }
255     return 1;
256 }
257
258 int insert(var d, ll *mem, int *pri, int *disp) {
259     int novo, p;
260     if (*disp == -1) return 0; //Nao cabe
261     novo = *disp; *disp = mem[*disp].link;
262     mem[novo].info = d;
263     p = hash(d.nome);
264     mem[novo].link = pri[p];
265     pri[p] = novo;
266     return 1;
267 }
268
269 int hash(char nome[]) {
270     int i = 0;
271     int soma = 0;
272     while (nome[i] != '\0') soma += (int)nome[i++];
273     return soma % HASH_LENGTH;
274 }
275
276 void reset_structs(ll *mem, int *pri, int *disp) {
277     int i;
278     var zero;
279     zero.tipo = 0;
280     zero.valor.i = 0;
281     strcpy(zero.nome, "");
282     for (i = 0; i < HASH_LENGTH; i++) pri[i] = -1;
283     for (i = 0; i < MAX_STRUCTS - 1; i++) { mem[i].link = i + 1; mem[i].info
    = zero; }
284     mem[i].link = -1;
285     mem[i].info = zero;
286     *disp = 0;
287 }
288
289 int busca(ll *mem, int *pri, char nome_busca[]) {
290     int p, x;
291     p = hash(nome_busca);
292     x = pri[p];
293     while (x != -1) {
294         if (!strcmp(mem[x].info.nome, nome_busca)) return x;
295         x = mem[x].link;
296     }
297     return -1;
298 }
299
300 void exibe(ll *mem, int *pri, int disp) {
301     int h;
302     for (h = 0; h < HASH_LENGTH; h++) {
303         printf("\t%2d - [%2d]\n", h, pri[h]);
304     }
305     printf("\n");
306     for (h = 0; h < MAX_STRUCTS; h++) {
307         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);
```

```
308         }
309         printf("\n");
310     }
```