```
#include <stdlib.h>
#include <stdio.h>
#include "mc_env.h"
#include "string.h"
#include "C.tab.h"
#include "genmc.h"

extern TOKEN * new_dst(FRME *);

TOKEN *lookup_loc(TOKEN * x, FRME * frame){
    while(frame != NULL){
        BNDING *bindings = frame->bindings;
        while(bindings != NULL){
            if(bindings->name == x){
                return bindings->loc;
            }
            bindings = bindings->next;
        }
        return NULL;
    }
}

TOKEN* lookup_reg(int x, FRME * frame){
    while(frame != NULL){
        BNDING *bindings = frame->bindings;
        while(bindings != NULL){
            if(bindings->loc != NULL && bindings->loc->value == x){
                return bindings->name;
            }
            bindings = bindings->next;
        }
        return NULL;
    }
}

void delete_loc(TOKEN * x, FRME * frame){
    while(frame != NULL){
        BNDING *bindings = frame->bindings;
        BNDING *head = bindings;
        BNDING *prev = NULL;
        while(bindings != NULL){
            if(bindings->name == x){
                if(prev == NULL){
                    frame->bindings = bindings->next;
                }
                else{
                    prev->next = bindings->next;
                    frame->bindings = head;
                }
                return;
            }
            prev = bindings;
            bindings = bindings->next;
        }
        frame = frame->next;
    }
}

void delete_constants(FRME* frame){
     while(frame != NULL){
```

```
61          BNDING *bindings = frame->bindings;
62          while(bindings != NULL){
63              if(bindings->name->type == CONSTANT){
64                  delete_loc(bindings->name,frame);
65              }
66              bindings = bindings->next;
67          }
68          frame = frame->next;
69      }
70  }
71
72  int reg_in_use(int x, FRME * frame){
73      while(frame != NULL){
74          BNDING *bindings = frame->bindings;
75          while(bindings != NULL){
76              if(bindings->loc != NULL && bindings->loc->value == x){
77                  return 1;
78              }
79              bindings = bindings->next;
80          }
81          frame = frame->next;
82      }
83      return 0;
84  }
85
86  TOKEN* use_temp_reg(FRME * frame){
87      TOKEN* t= new_dst(frame);
88      if(t == NULL ) {printf("error: all registers in use!");exit(1);}
89      BNDING *bindings = frame->bindings;
90      BNDING *new = malloc(sizeof(BNDING));
91      if(new != NULL){
92          new->type = IDENTIFIER;
93          new->loc = t;
94          new->next = bindings;
95          frame->bindings=new;
96          return t;
97      }
98      printf("fatal: binding creation failed!\n");
99  }
100
101 TOKEN *assign_to_var(TOKEN * x, FRME * frame, TOKEN* loc){
102     while(frame != NULL){
103         BNDING *bindings = frame->bindings;
104         while(bindings != NULL){
105             if(bindings->name == x){
106                 if(reg_in_use(loc->value,frame)){
107                     delete_loc(lookup_reg(loc->value,frame),frame);
108                 }
109                 bindings->loc = loc;
110                 return loc;
111             }
112             bindings = bindings->next;
113         }
114         frame = frame->next;
115     }
116     printf("fatal: unbound variable!\n");exit(1);
117 }
118
119 void declare_var(TOKEN * x, FRME * frame){
120     BNDING *bindings = frame->bindings;
```

```
121     BNDING *new = malloc(sizeof(BNDING));
122     if(new != NULL){
123         new->type = IDENTIFIER;
124         new->name = x;
125         new->loc = NULL;
126         new->next = bindings;
127         frame->bindings=new;
128         return;
129     }
130     printf("fatal: binding creation failed!\n");
131 }
132 
133 TOKEN *declare_fnc(TOKEN * x, CLSURE* val, FRME * frame){
134     BNDING *bindings = frame->bindings;
135     BNDING *new = malloc(sizeof(BNDING));
136     if(new != NULL){
137         new->type = CLOS;
138         new->name = x;
139         new->clos = val;
140         new->next = bindings;
141         frame->bindings=new;
142         return new->name;
143     }
144     printf("fatal: binding creation failed!\n");
145 }
146 
147 CLSURE *find_fnc(TOKEN* name, FRME* e){
148     FRME *ef = e;
149     BNDING* bindings;
150      while(ef != NULL){
151         bindings = ef->bindings;
152         while (bindings != NULL){
153             if(bindings->name ==  name){
154                return bindings->clos;
155             }
156             bindings = bindings->next;
157         }
158          ef = ef->next;
159      }
160      return NULL;
161 }
162 
163
```