TI DSP, MCU, Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee (이상훈)
gcccompil3r@gmail.com
학생 – 김형주
mihaelkel@naver.com

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
typedef struct __stack
{
    char eq[64];
    struct __stack* node;
}stack;
typedef struct __expr
{
    char eq[64];
    struct __expr* left;
    struct __expr* right;
}expr;
expr* get_expr_node(void);
void ins_expr_node(expr** root);
int find_equal_idx(char* buf);
void init_expr_node(expr** root, expr** ln, expr** rn);
void set_ln_node(expr** ln, char* buf, int eq_idx);
void set_rn_node(expr** rn, char* buf, int eq_idx);
stack* get_stack_node(void);
void push(stack** top, char* buf);
void pop(stack** top,char* buf);
bool is_not_empty(stack** top);
void ins_tree(expr** root, char* buf);
void print_tree(expr** root);
int main(void)
{
    /*root must have '='*/
    expr* root = NULL;
    /*ln means left node, should be relative with y or y', without x*/
    expr* ln = NULL;
    /*rn means right node, should be relative with x and constant number, without y*/
    expr* rn = NULL;
    /*buf has the equation, f.e. y'=-2xy*/
    char buf[64] = "y'=-2xy";
    /*eq_idx means equal index, This indicates where '=' is.*/
    int eq_idx;
    /*set eq_idx*/
    eq_idx = find_equal_idx(buf);
    /*set initial tree. root get value '='*/
    init_expr_node(&root,&ln,&rn);
    set_ln_node(&ln, buf, eq_idx);
    set_rn_node(&rn, buf, eq_idx);
    printf("root : %s\n",root->eq);
```

```
47      printf("left : %s\n",root->left->eq);
48      printf("right : %s\n",root->right->eq);
49
50      ins_expr_node(&ln);
51      print_tree(&root);
52      printf("test : %s\n",root->left->left->eq);
53  }
54  expr* get_expr_node(void)
55  {
56      expr* tmp;
57      tmp = (expr*)malloc(sizeof(expr)*1);
58      tmp->left = NULL;
59      tmp->right = NULL;
60      return tmp;
61  }
62  void ins_expr_node(expr** root)
63  {
64      char buf[32];
65      char tmp_buf[32];
66      char tmp;
67      int i = 0;
68      stack* top = NULL;
69      while((*root)->eq[i])
70      {
71          tmp = (*root)->eq[i];
72          /*tmp is number(0~9) or 'x' or 'y'*/
73          if( tmp=='x' || tmp == 'y' || ((48 <=tmp) &&(tmp <= 57)) )
74              push(&top, &tmp);
75          else
76          {
77              sprintf(tmp_buf,"%c",tmp);
78              strcat(buf, tmp_buf);
79              ins_tree(root,tmp_buf);
80          }
81          i++;
82      }
83      while(is_not_empty(&top))
84      {
85          pop(&top, buf);
86          ins_tree(root,buf);
87      }
88  }
89  /*find where '=' is*/
90  int find_equal_idx(char* buf)
91  {
92      int i;
```

```
    for(i=0;buf[i];i++)
        if(buf[i] == '=')
            break;
    return i;
}
void init_expr_node(expr** root, expr** ln, expr** rn)
{
    *root = get_expr_node();
    *ln   = get_expr_node();
    *rn      = get_expr_node();
    (*root)->left  = *ln;
    (*root)->right = *rn;

    strcat((*root)->eq,"=");
}
void set_ln_node(expr** ln, char* buf, int eq_idx)
{
    strncpy((*ln)->eq, buf, eq_idx);
}
void set_rn_node(expr** rn, char* buf, int eq_idx)
{
    strncpy((*rn)->eq, buf + eq_idx + 1, strlen(buf) - eq_idx - 1);
}
stack* get_stack_node(void)
{
    stack* tmp;
    tmp = (stack*)malloc(sizeof(stack)*1);
    tmp->node = NULL;
    return tmp;
}
void push(stack** top, char* buf)
{
    stack* tmp = *top;
    *top = get_stack_node();
    strcat((*top)->eq, buf);
    (*top)->node = tmp;
}
void pop(stack** top,char* buf)
{
    if(!(*top)->eq)
    {
        strcat(buf,"");
        return;
    }
    stack* tmp = *top;
    strcpy(buf, (*top)->eq);
```

```
139
140     *top = (*top)->node;
141     free(tmp);
142 }
143 bool is_not_empty(stack** top)
144 {
145     if(*top)
146         return true;
147     return false;
148 }
149 void ins_tree(expr** root, char* buf)
150 {
151     expr* tmp = *root;
152     while(tmp)
153     {
154         if(!tmp->left)
155             tmp = tmp->left;
156         else if(!tmp->right)
157             tmp = tmp->right;
158         else
159             tmp = tmp->left;
160     }
161     tmp = get_expr_node();
162     strcpy(tmp->eq, buf);
163 }
164 void print_tree(expr** root)
165 {
166     expr* tmp = *root;
167     if(tmp)
168     {
169         printf("value : %s\n",tmp->eq);
170         print_tree(&tmp->left);
171         print_tree(&tmp->right);
172     }
173     return;
174 }
175
```