06. Trees

dbserver.korea@gmail.com

Agenda

- Tree Problem
  - Expression Tree ADT

Folder> 1_TreeADT

Expression Tree ADT

- Implement an Expression Tree ADT
- Operations :
    - **createTree** - create a tree
    - **destroyTree** - delete a tree
    - **treeData** - return a data in the root node
    - **hasChild** - return false if a node has no child
    - **evaluate** - calculate expression and return the result

Type and Functions for Expression Tree ADT

```c
#pragma once
#include <stdbool.h>
#define STR_MAX 16

typedef struct node
{
	char dataPtr[STR_MAX];
	struct node* left;
	struct node* right;
} NODE;

NODE* createTree(NODE* left, const char* dataPtr, NODE* right);
void destroyTree(NODE* node);
char* treeData(NODE* node);

bool hasChild(NODE* node);
double evaluate(NODE* node);
```

Expression Tree ADT - Main function

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "expressionTreeADT.h"

int main()
{
     NODE* left, *right, *root;

     // -> 6 / 2
     left = createTree(NULL, "6", NULL);
     right = createTree(NULL, "2", NULL);
     root = createTree(left, "/", right);

     // -> 3 + ( )
     left = createTree(NULL, "3", NULL);
     right = root;
     root = createTree(left, "+", right);

     // -> 2 * ( )
     left = createTree(NULL, "2", NULL);
     right = root;
     root = createTree(left, "*", right);

     // -> ( ) / 4
     left = root;
     right = createTree(NULL, "4", NULL);
     root = createTree(left, "/", right);

     if (hasChild(root))
     {
          printf("Result: %f\n", evaluate(root));
     }

     destroyTree(root);

     return 0;
}
```