第1章《表达式解释》实验报告

21051117 卢俊成

主要数据结构和变量

- 栈 `Stack` 及其相关操作函数

程序主要流程

与题目提示一致

程序主要函数功能

- 栈 `Stack` 相关操作函数
- `int priority(char)`
  返回运算符优先级
- `int operate(int, int, char)`
  实际运算并返回

已实现功能

全部题目要求功能

编译与运行信息

编译信息

```
/usr/bin/gcc -std=c17 -fdiagnostics-color=always -Wfatal-errors -Wall -Wextra -g
1.c -o 1 -lm
```

测试数据1

输入

```
((2+3)*6/8+9*20)#
```

输出

```
183
```

测试数据2

输入

```
#
```

输出

```
0
```

测试数据3

输入

```
1+2+3+4+5+6+7+8+9+10#
```

输出

```
55
```

测试数据4

输入

```
100-9-8-7-6-5-4-3-2-1#
```

输出

```
55
```

测试数据5

输入

```
2*3*5*7#
```

输出

```
210
```

测试数据6

输入

```
441/5/7#
```

输出

```
12
```

测试数据7

输入

```
20*4+15/9#
```

输出

```
81
```

测试数据8

输入

```
()#
```

输出

```
0
```

测试数据9

输入

```
1/0#
```

输出

```
Divide by zero.
```

测试数据10

输入

```
8*2/9*9-1/8+5/8*5#
```

输出

```
9
```

源代码

```
// Work on stdc17
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

typedef struct node {
    int data;
    struct node* next;
} Node;

typedef struct Stack {
    int length;
    Node* head;
} Stack;

Stack* Create() {
    Stack* s = (Stack*)malloc(sizeof(Stack));
    if (s == NULL) {
        fprintf(stderr, "Error for malloc.\n");
        exit(-1);
    }
    s->length = 0;
    s->head = NULL;
    return s;
}
```

```
void Destroy(Stack* s) {
    Node* current = s->head;
    Node* tmp;
    while (current != NULL) {
        tmp = current;
        current = current->next;
        free(tmp);
    }
    free(s);
}

int IsEmpty(Stack* s) {
    return s->length == 0;
}

int getTop(Stack* s) {
    if (IsEmpty(s)) {
        fprintf(stderr, "Stack is empty.\n");
        exit(-1);
    }
    return s->head->data;
}

void pop(Stack* s) {
    if (IsEmpty(s)) {
        fprintf(stderr, "Stack is empty.\n");
        exit(-1);
    }
    Node* tmp = s->head;
    s->head = s->head->next;
    free(tmp);
    s->length--;
}

void push(Stack* s, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (newNode == NULL) {
        fprintf(stderr, "Error for malloc.\n");
        exit(-1);
    }
    newNode->data = data;
    newNode->next = s->head;
    s->head = newNode;
    s->length++;
}

int priority(char c) {
    if (c == ')') return 3;
    if (c == '*' || c == '/') return 2;
    if (c == '+' || c == '-') return 1;
    return 0;
}

int operate(int a, int b, char c) {
```

```
    if (c == '+') return a + b;
    if (c == '-') return a - b;
    if (c == '*') return a * b;
    if (c == '/') {
        if (b == 0) {
            fprintf(stderr, "Divide by zero.\n");
            exit(-1);
        }
        return a / b;
    }
    return 0;
}

int main() {
#ifndef ONLINE_JUDGE
    freopen("1.in", "r", stdin);
    freopen("1.out", "w", stdout);
    freopen("1.err", "w", stderr);
#endif
    Stack* num = Create();
    Stack* op = Create();
    char ch;
    int n, num1, num2;
    push(op, '(');
    while ((ch = getchar())) {
        if (isdigit(ch)) {
            n = ch - '0';
            while (isdigit(ch = getchar()))
                n = n * 10 + ch - '0';
            push(num, n);
        }
        if (ch == '(') {
            push(op, ch);
            continue;
        }
        if (ch == ')') {
            while (getTop(op) != '(') {
                num1 = getTop(num);
                pop(num);
                num2 = getTop(num);
                pop(num);
                push(num, operate(num2, num1, getTop(op)));
                pop(op);
            }
            pop(op);
            continue;
        }
        if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
            if (IsEmpty(op) || (priority(ch) > priority(getTop(op)))) {
                push(op, ch);
                continue;
            }
            num1 = getTop(num);
            pop(num);
            num2 = getTop(num);
```

```
            pop(num);
            push(num, operate(num2, num1, getTop(op)));
            pop(op);
            push(op, ch);
            continue;
        }
        if (ch == '#') break;
    }
    while (getTop(op) != '(') {
        num1 = getTop(num);
        pop(num);
        num2 = getTop(num);
        pop(num);
        push(num, operate(num2, num1, getTop(op)));
        pop(op);
    }
    if (IsEmpty(num)) {
        printf("0\n");
    } else {
        printf("%d\n", getTop(num));
    }
    Destroy(num);
    Destroy(op);
#ifndef ONLINE_JUDGE
    fclose(stdin);
    fclose(stdout);
    fclose(stderr);
#endif
    return 0;
}
```