

# 자료구조론

## Homework # 3

이름 : 윤성호

학번 : 12161756

분반 : 004

제출일 : 2019/05/11

### - 개요 -

#### **(1) Doubly Linked List**

1. 구현상 특징 : p2
2. 실행 화면 : p3

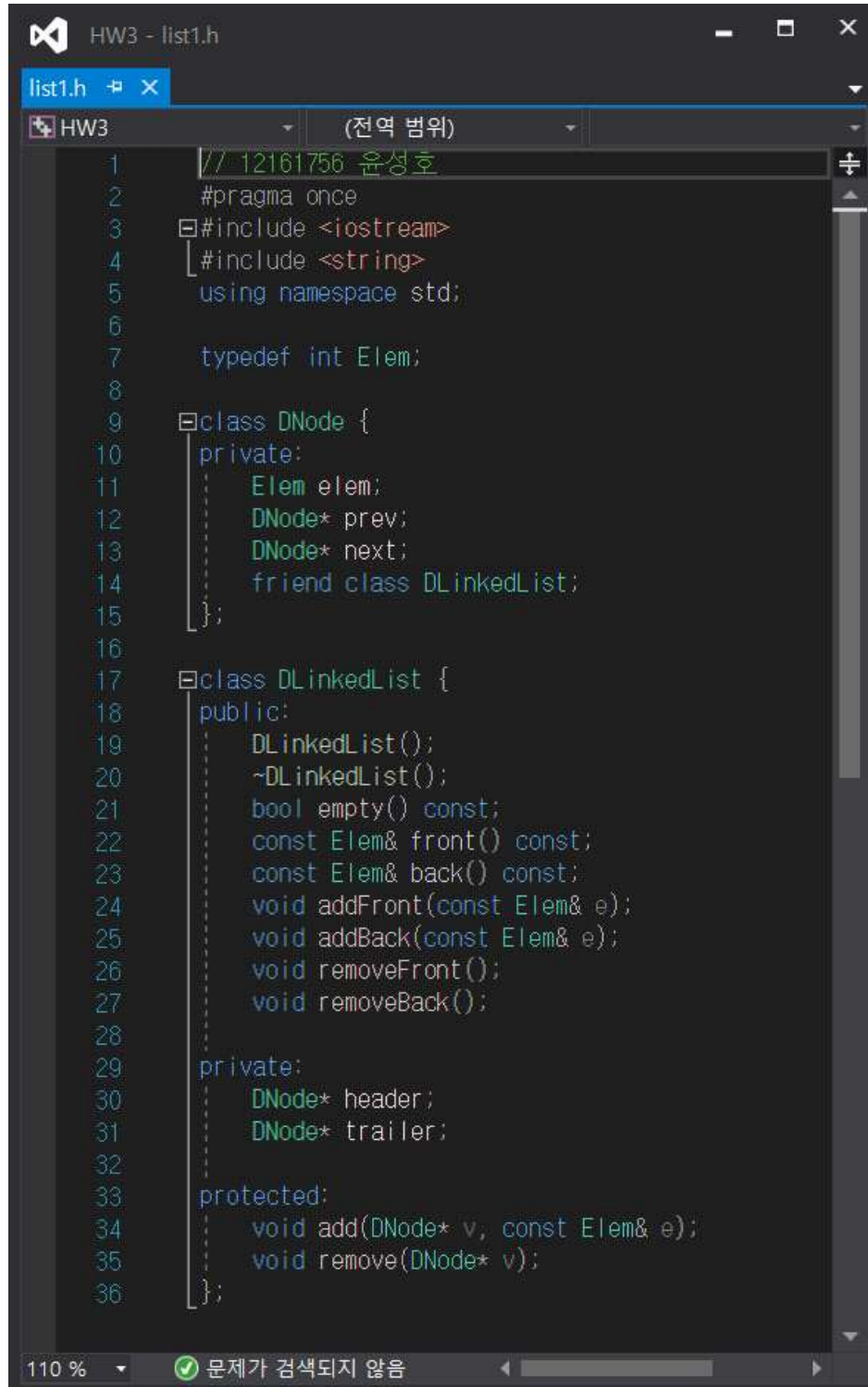
#### **(2) Parentheses Matching**

1. 구현상 특징 : p4~6
2. 실행 화면 : p7

## (1) Doubly Linked List

### 1. 구현상 특징

과제 지시사항과 같이 책의 코드를 활용하였고, 책과 다른 점이라면 `typedef string Elem -> typedef int Elem` 입니다. 과제에서 임의의 정수로 동작함을 보이려하여 `int` 로 고쳐줬습니다.



```

1 // 12161756 윤성호
2 #pragma once
3 #include <iostream>
4 #include <string>
5 using namespace std;
6
7 typedef int Elem;
8
9 class DNode {
10 private:
11     Elem elem;
12     DNode* prev;
13     DNode* next;
14     friend class DLinkedList;
15 };
16
17 class DLinkedList {
18 public:
19     DLinkedList();
20     ~DLinkedList();
21     bool empty() const;
22     const Elem& front() const;
23     const Elem& back() const;
24     void addFront(const Elem& e);
25     void addBack(const Elem& e);
26     void removeFront();
27     void removeBack();
28
29 private:
30     DNode* header;
31     DNode* trailer;
32
33 protected:
34     void add(DNode* v, const Elem& e);
35     void remove(DNode* v);
36 };
  
```

## 2. 실행 화면

main.cpp 구현은 아래와 같으며, 잘 실행되는 모습입니다.

```

1 // 12161756 윤성호
2 #include "list1.h"
3
4 int main() {
5     cout << "12161756 윤성호" << endl;
6     DLinkedList ex;
7
8     for (int i = 1; i <= 5; i++) {
9         ex.addBack(i);
10    }
11    // 초기 순서 : 1, 2, 3, 4, 5
12
13    cout << "front() 결과 : " << ex.front() << endl; // 1
14    cout << "back() 결과 : " << ex.back() << endl; // 5
15
16    ex.removeBack(); // 5 제거
17    ex.removeFront(); // 1 제거
18
19    cout << endl;
20    cout << "2. removeBack과 removeFront 한 번씩 실행한 후" << endl;
21    cout << "front() 결과 : " << ex.front() << endl; // 2
22    cout << "back() 결과 : " << ex.back() << endl; // 4
23
24    // 최종 순서 : 2, 3, 4
25
26    return 0;
27 }
  
```

```

Microsoft Visual Studio 디버그 콘솔
12161756 윤성호
front() 결과 : 1
back() 결과 : 5

2. removeBack과 removeFront 한 번씩 실행한 후
front() 결과 : 2
back() 결과 : 4

C:\Users\grymp\Desktop\Project\HW3\Debug\HW3.exe
(12876 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
  
```

## (2) Parentheses Matching

### 1. 구현상 특징

```

1 // 12161756 윤성호
2 #pragma once
3 #include <iostream>
4 #include <string>
5 #include <list>
6 using namespace std;
7 typedef char Elem;
8
9 bool parCheck(char* str); // DLinkStack으로 구현한 괄호 검사 함수
10 bool STLparCheck(char* str); // STL로 구현한 괄호 검사 함수
11 void getMsg(bool TorF); // 괄호 검사 함수 결과를 메시지로 출력하는 함수
12
13 class DNode {
14 private:
15     Elem elem;
16     DNode* prev;
17     DNode* next;
18     friend class DLinkStack;
19 };
20
21 class DLinkStack{ // DLinklist에서 Stack에 활용될 수 있는 함수만 뽑아
22 public: // 이름만 Stack 함수명에 맞게 바꿔준 클래스
23     DLinkStack();
24     ~DLinkStack();
25     bool empty() const;
26     const Elem& top() const;
27     void push(const Elem& e);
28     void pop();
29
30 private:
31     DNode* header;
32     DNode* trailer;
33
34 protected:
35     void add(DNode* v, const Elem& e);
36     void remove(DNode* v);
37 };

```

우선 과제요구사항에 맞게 (1)에서 작성한 코드 중 **Stack** 구현에 필요한 함수만 뽑아 **Stack**에 사용되는 함수명에 맞게 이름만 바꿔 클래스를 재구성하였습니다.

또한 괄호 검사 구현에 필요한 3 가지 함수를 작성했습니다.

```

114 void getMsg(bool TorF) {
115     if (TorF == true) {
116         cout << "correct" << endl; // 넘겨받은 값이 true면 "correct" 출력
117     }
118
119     else
120         cout << "incorrect" << endl; // 넘겨받은 값이 false면 "incorrect" 출력
121 }
122
123

```

괄호 검사 함수의 반환 값을 이용해 메시지를 출력하는 함수입니다.

```

1 // 12161756 윤성호
2 #include "list2.h"
3
4 bool parCheck(char* str) {
5     DLinkedStack s1;           // DLinkedStack s1 생성
6     string s = str;           // 넘겨 받은 char형 문자열을 string으로 변환
7     int length = s.length();   // 변환한 string의 길이 저장
8
9     for(int i = 0; i < length; i++) {
10        if (str[i] == '(' || str[i] == '{' || str[i] == '[')
11        {
12            s1.push(str[i]);
13        }
14        // 괄호 종류 관계없이 여는 괄호면 push
15
16        else if (str[i] == ')') {
17            if (s1.empty())
18                return false;    // 닫는 소괄호이면서, stack이 비어있다면 false
19
20            else {
21                if (s1.top() == '(')
22                    s1.pop();    // 닫는 소괄호이면서, top이 여는 소괄호면 pop
23                else
24                    return false; // 닫는 소괄호이지만,
25                                // top이 여는 소괄호가 아니면 false
26            }
27
28            else if (str[i] == '}') { // 하단의 중괄호, 대괄호 또한
29                if (s1.empty())      // 위의 소괄호 코드와 동일하게 구성하였습니다.
30                    return false;
31
32                else {
33                    if (s1.top() == '{')
34                        s1.pop();
35                    else
36                        return false;
37                }
38
39            else if (str[i] == ']') {
40                if (s1.empty())
41                    return false;
42
43                else {
44                    if (s1.top() == '[')
45                        s1.pop();
46                    else
47                        return false;
48                }
49            }
50            // 문자열 길이만큼 for구문 탐색 완료
51
52            if (s1.empty())
53                return true;    // for구문이 완료됐을때, stack이 비어있다면 true
54
55            else
56                return false;   // 완료됐는데 stack에 남은 괄호가 있다면 false
57        }
58    }

```

```

HW3-2 - list2.cpp
list2.cpp
HW3-2
(전역 범위)
parCheck(char * str)

59 bool STLparCheck(char* str) {
60     list<char> s2; // char형 list s2 생성
61     string s = str; // 넘겨 받은 char형 문자열을 string으로 변환
62     int length = s.length(); // 변환한 string의 길이 저장
63
64     for (int i = 0; i < length; i++) {
65         if (str[i] == '(' || str[i] == '[' || str[i] == '{')
66         {
67             s2.push_front(str[i]);
68         } // 괄호 종류 관계없이 여는 괄호면 push_front
69
70         else if (str[i] == ')') {
71             if (s2.empty())
72                 return false; // 닫는 소괄호이면서, stack이 비어있다면 false
73
74             else {
75                 if (s2.front() == '(')
76                     s2.pop_front(); // 닫는 소괄호이면서, front가 여는 소괄호면 pop_front
77                 else
78                     return false; // 닫는 소괄호이지만,
79                                     // front가 여는 소괄호가 아니면 false
80             }
81
82             else if (str[i] == ']') {
83                 if (s2.empty()) // 하단의 중괄호, 대괄호 또한
84                     return false; // 위의 소괄호 코드와 동일하게 구성하였습니다.
85
86                 else {
87                     if (s2.front() == '[')
88                         s2.pop_front();
89                     else
90                         return false;
91                 }
92             }
93         }
94
95         else if (str[i] == '}') {
96             if (s2.empty())
97                 return false;
98
99             else {
100                 if (s2.front() == '[')
101                     s2.pop_front();
102                 else
103                     return false;
104             }
105         } // 문자열 길이만큼 for구문 탐색 완료
106
107         if (s2.empty())
108             return true; // for구문이 완료됐을때, stack이 비어있다면 true
109
110         else
111             return false; // 완료됐는데 stack에 남은 괄호가 있다면 false
112     }
113 }

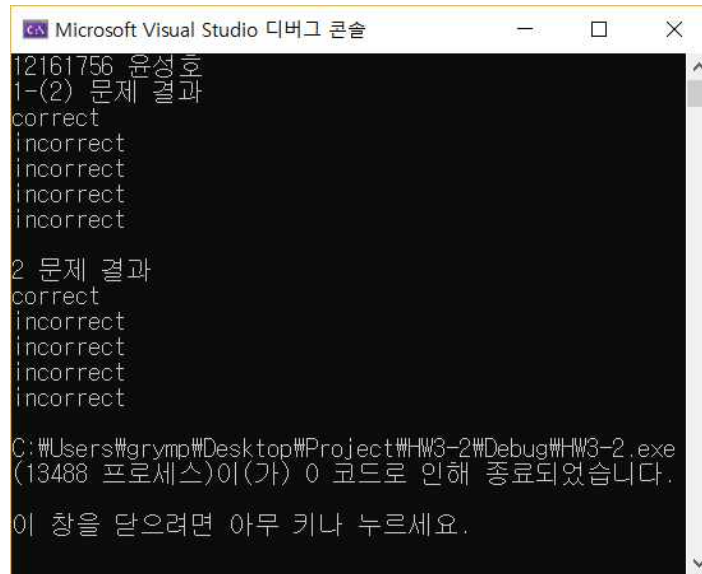
```

p5 에 있는 것은 Doubly Linked list 로 구현한 스택을 기반으로 작성한 괄호 검사 함수이며  
p6 에 있는 것은 STL 의 list 를 이용해 구현한 괄호 검사 함수입니다.



## 2. 실행 화면

문제없이 잘 실행되고 있는 모습입니다.



```

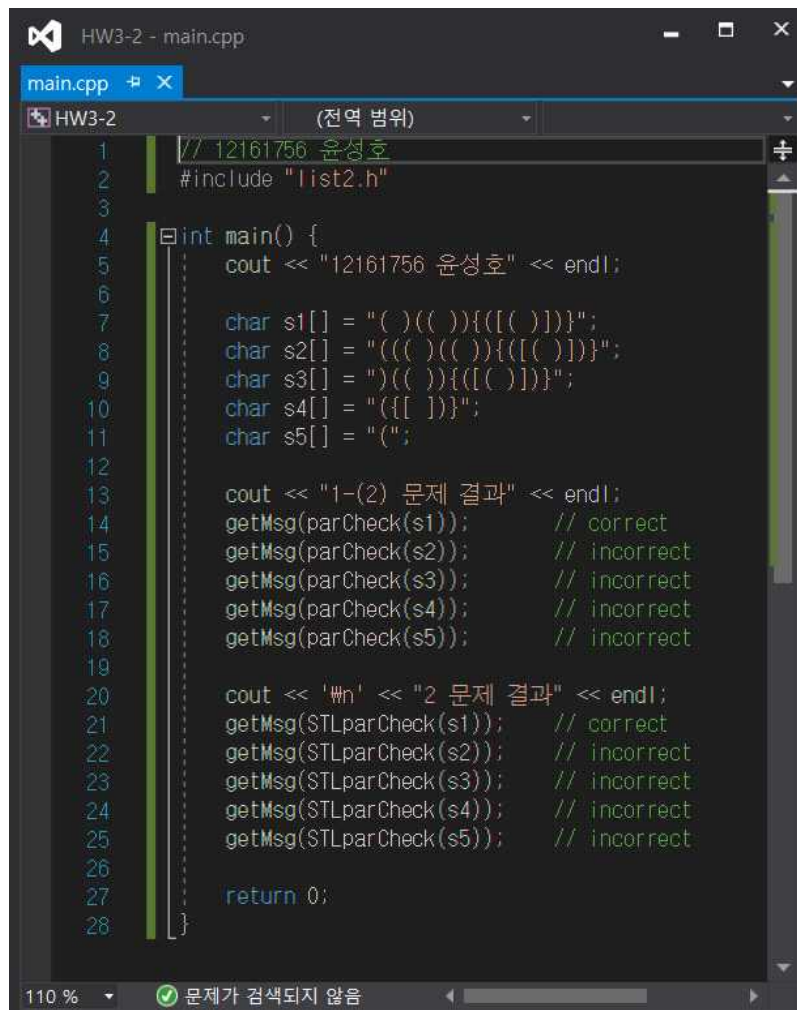
Microsoft Visual Studio 디버그 콘솔
12161756 윤성호
1-(2) 문제 결과
correct
incorrect
incorrect
incorrect
incorrect

2 문제 결과
correct
incorrect
incorrect
incorrect
incorrect

C:\Users#grymp\Desktop\Project\HW3-2\Debug\HW3-2.exe
(13488 프로세스)이(가) 0 코드로 인해 종료되었습니다.

이 창을 닫으려면 아무 키나 누르세요.

```



```

HW3-2 - main.cpp
main.cpp
HW3-2 (전역 범위)
1 // 12161756 윤성호
2 #include "list2.h"
3
4 int main() {
5     cout << "12161756 윤성호" << endl;
6
7     char s1[] = "( )(( )){([ ( ))}";
8     char s2[] = "((( )(( )){([ ( ))}";
9     char s3[] = ")(( )){([ ( ))}";
10    char s4[] = "([ ]}";
11    char s5[] = "(";
12
13    cout << "1-(2) 문제 결과" << endl;
14    getMsg(parCheck(s1)); // correct
15    getMsg(parCheck(s2)); // incorrect
16    getMsg(parCheck(s3)); // incorrect
17    getMsg(parCheck(s4)); // incorrect
18    getMsg(parCheck(s5)); // incorrect
19
20    cout << '\n' << "2 문제 결과" << endl;
21    getMsg(STLparCheck(s1)); // correct
22    getMsg(STLparCheck(s2)); // incorrect
23    getMsg(STLparCheck(s3)); // incorrect
24    getMsg(STLparCheck(s4)); // incorrect
25    getMsg(STLparCheck(s5)); // incorrect
26
27    return 0;
28 }
110 % 문제 검색되지 않음

```