

컴퓨터알고리즘과실습 2주차 실습

2020112046 이재원

2 - 1.

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4
5  using namespace std;
6
7  class Stack1 {
8  private:
9      int *stack;
10     int p;
11
12 public:
13     Stack1(int max = 100) {
14         stack = new int[max];
15         p = 0;
16     }
17     ~Stack1() {
18         delete stack;
19     }
20     inline void push(int x) {
21         stack[p++] = x;
22     }
23     inline int pop() {
24         return stack[--p];
25     }
26     inline int empty() {
27         return !p;
28     }
29 };
30
31 int main() {
32     char c;
33     Stack1 acc(50);
34     int x;
35
36     while ((c = cin.get()) != '\n') {
37         x = 0;
38         while (c == ' ') {
39             cin.get(c);
40         }
41         if (c == '+') {
42             x = acc.pop() + acc.pop();
43         }
44         if (c == '*') {
45             x = acc.pop() * acc.pop();
46         }
47         while (c >= '0' && c <= '9') {
48             x = x * 10 + (c - '0');
49             cin.get(c);
50         }
51         acc.push(x);
52     }
53     cout << acc.pop() << endl;
54 }
55
```

```

#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Stack1 {
private:
    int *stack;
    int p;

public:
    Stack1(int max = 100) {
        stack = new int[max];
        p = 0;
    }
    ~Stack1() {
        delete stack;
    }
    inline void push(int x) {
        stack[p++] = x;
    }
    inline int pop() {
        return stack[--p];
    }
    inline int empty() {
        return !p;
    }
};

int main() {
    char c;
    Stack1 acc(50);
    int x;

    while ((c = cin.get()) != '\n') {
        x = 0;
        while (c == ' ') {
            cin.get(c);
        }
        if (c == '+') {
            x = acc.pop() + acc.pop();
        }
        if (c == '*') {
            x = acc.pop() * acc.pop();
        }
        while (c >= '0' && c <= '9') {
            x = x * 10 + (c - '0');
            cin.get(c);
        }
        acc.push(x);
    }
    cout << acc.pop() << endl;
}

```

1번 문제는 배열을 이용해 Stack을 구현하고, 후위 표기법(postfix)으로 된 식을 계산해보는 간단한 문제였다.

push, pop 모두 $O(1)$ 의 시간이 걸리기에, 전체 시간 복잡도는 입력된 숫자 or 연산자의 수를 n 이라 할 때 , $O(n)$.

Space Complexity도 $O(n)$.

```
25     }
26     inline int empty() {
27     |     return !p;
28     }
29 };
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g test.cpp -o a.out && "/Users/jaewonlee/Desktop/PS/"a.out
● > cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g test.cpp -o a.out && "/Users/jaewonlee/Desktop/PS/"a.out
20 30 * 45 3 * + 10 * 9 +
7359
```

2 – 2.

```

1  #include <iostream>
2
3  using namespace std;
4
5  class Stack2 {
6  private:
7      struct node {
8          int data;
9          node *next;
10         node *prev;
11     };
12     struct node *head, *cursor;
13
14 public:
15     Stack2(int max = 100) {
16         head = NULL;
17         cursor = NULL;
18     }
19     ~Stack2() {
20         head = NULL;
21         cursor = NULL;
22     }
23     void push(int x) {
24         node *temp = new node;
25         temp->data = x;
26         temp->next = NULL;
27         temp->prev = NULL;
28         if (head == NULL) {
29             head = temp;
30             cursor = temp;
31         } else {
32             cursor->next = temp;
33             temp->prev = cursor;
34             cursor = temp;
35         }
36     }
37     int pop() {
38         if (head == NULL) {
39             cout << "Stack is empty" << endl;
40             return 0;
41         } else {
42             int x = cursor->data;
43             if (cursor == head) {
44                 delete cursor;
45                 cursor = NULL;
46                 head = NULL;
47             } else {
48                 cursor = cursor->prev;
49                 delete cursor->next;
50                 cursor->next = NULL;
51             }
52             return x;
53         }
54     }
55     void empty() {
56         while (head != NULL) {
57             pop();
58         }
59     }
60 };
61
62 int main() {
63     char c;
64     Stack2 acc(50);
65     int x, y;
66
67     while ((c = cin.get()) != '\n') {
68         x = 0;
69         while (c == ' ') {
70             cin.get(c);
71         }
72         if (c == '+') {
73             x = acc.pop() + acc.pop();
74         }
75         if (c == '*') {
76             x = acc.pop() * acc.pop();
77         }
78         if (c == '-') {
79             x = acc.pop();
80             y = acc.pop();
81             x = y - x;
82         }
83         if (c == '/') {
84             x = acc.pop();
85             y = acc.pop();
86             x = y / x;
87         }
88         while (c >= '0' && c <= '9') {
89             x = x * 10 + (c - '0');
90             cin.get(c);
91         }
92         acc.push(x);
93     }

```

```

#include <iostream>

using namespace std;

class Stack2 {
private:
    struct node {
        int data;
        node *next;
        node *prev;
    };
    struct node *head, *cursor;

public:
    Stack2(int max = 100) {
        head = NULL;
        cursor = NULL;
    }
    ~Stack2() {
        head = NULL;
        cursor = NULL;
    }
    void push(int x) {
        node *temp = new node;
        temp->data = x;
        temp->next = NULL;
        temp->prev = NULL;
        if (head == NULL) {
            head = temp;
            cursor = temp;
        } else {
            cursor->next = temp;
            temp->prev = cursor;
            cursor = temp;
        }
    }
    int pop() {
        if (head == NULL) {
            cout << "Stack is empty" << endl;
            return 0;
        } else {
            int x = cursor->data;
            if (cursor == head) {
                delete cursor;
                cursor = NULL;
                head = NULL;
            } else {
                cursor = cursor->prev;
                delete cursor->next;
                cursor->next = NULL;
            }
            return x;
        }
    }
}

```

```

    void empty() {
        while (head != NULL) {
            pop();
        }
    }
};

int main() {
    char c;
    Stack2 acc(50);
    int x, y;

    while ((c = cin.get()) != '\n') {
        x = 0;
        while (c == ' ') {
            cin.get(c);
        }
        if (c == '+') {
            x = acc.pop() + acc.pop();
        }
        if (c == '*') {
            x = acc.pop() * acc.pop();
        }
        if (c == '-') {
            x = acc.pop();
            y = acc.pop();
            x = y - x;
        }
        if (c == '/') {
            x = acc.pop();
            y = acc.pop();
            x = y / x;
        }
        while (c >= '0' && c <= '9') {
            x = x * 10 + (c - '0');
            cin.get(c);
        }
        acc.push(x);
    }
    cout << acc.pop() << endl;
}

```

2번 문제는 1번 문제와 동일하게 후위 표기법으로 쓰인 연산식을 계산하는 문제였는데,

Stack을 포인터를 이용하여 구현해야 했다.

Singly Linked List로 Stack을 구현하면,

pop 과정에서 새로운 tail을 찾을 때마다 $O(n)$ 의 시간이 걸리므로,

push와 pop 모두 $O(1)$ 로 끝낼 수 있는 Doubly Linked List로 Stack을 구현했다.

그리하여 1번과 동일하게 전체 시간 복잡도는 $O(n)$.

당연히 Space Complexity도 $O(n)$.

```
88         while (c >= '0' && c <= '9') {
89             x = x * 10 + (c - '0');
90             cin.get(c);
91         }
92         acc.push(x);
93     }
94     cout << acc.pop() << endl;
95 }
96
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
> cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g test.cpp -o a.out && "/Users/
jaewonlee/Desktop/PS/"a.out
20 30 * 45 3 / + 10 * 9 -
6141
```


2 - 3.

```
1 #include <stdlib.h>
2 #include <time.h>
3
4 #include <deque>
5 #include <iostream>
6 #include <vector>
7
8 using namespace std;
9
10 int main() {
11     srand((unsigned)time(NULL));
12
13     // Create a vector of Union and Intersection of A, B
14     vector<int> U;
15     vector<int> X;
16
17     // n, m >= 10
18     int n, m;
19     cin >> n >> m;
20
21     deque<int> A;
22     deque<int> B;
23
24     // number : 1 ~ 10 - 집합이 아니라 집단이라고 했으므로, 중복된 값이 들어갈 수 있다.
25     while (A.size() < n) {
26         A.push_back(rand() % 10 + 1);
27     }
28     while (B.size() < m) {
29         B.push_back(rand() % 10 + 1);
30     }
31
32     // union : A + B
33     for (int i = 0; i < A.size(); i++) {
34         auto iter = find(U.begin(), U.end(), A[i]);
35         if (iter == U.end()) {
36             U.push_back(A[i]);
37         }
38     }
39     for (int i = 0; i < B.size(); i++) {
40         auto iter = find(U.begin(), U.end(), B[i]);
41         if (iter == U.end()) {
42             U.push_back(B[i]);
43         }
44     }
45
46     // intersection : A * B
47     for (int i = 0; i < A.size(); i++) {
48         auto iter = find(B.begin(), B.end(), A[i]);
49         auto iter2 = find(X.begin(), X.end(), A[i]);
50         if (iter != B.end() && iter2 == X.end()) {
51             X.push_back(A[i]);
52         }
53     }
54
55     // print A, B
56     cout << "A : ";
57     for (int i = 0; i < A.size(); i++) {
58         cout << A[i] << " ";
59     }
60     cout << endl;
61     cout << "B : ";
62     for (int i = 0; i < B.size(); i++) {
63         cout << B[i] << " ";
64     }
65     cout << endl;
66
67     // print vector U
68     cout << "U = { ";
69     for (int i = 0; i < U.size(); i++) {
70         cout << U[i] << " ";
71     }
72     cout << "}" << endl;
73     cout << "X = { ";
74     // print vector X
75     for (int i = 0; i < X.size(); i++) {
76         cout << X[i] << " ";
77     }
78     cout << "}" << endl;
79 }
80
```

#include <stdlib.h>

```

#include <time.h>

#include <deque>
#include <iostream>
#include <vector>

using namespace std;

int main() {
    srand((unsigned)time(NULL));

    // Create a vector of Union and Intersection of A, B
    vector<int> U;
    vector<int> X;

    // n, m >= 10
    int n, m;
    cin >> n >> m;

    deque<int> A;
    deque<int> B;

    // number : 1 ~ 10000 - 집합이 아니라 집단이라고 했으므로, 중복된 값이 들어갈 수 있다.
    while (A.size() < n) {
        A.push_back(rand() % 10 + 1);
    }
    while (B.size() < m) {
        B.push_back(rand() % 10 + 1);
    }

    // union : A + B
    for (int i = 0; i < A.size(); i++) {
        auto iter = find(U.begin(), U.end(), A[i]);
        if (iter == U.end()) {
            U.push_back(A[i]);
        }
    }
    for (int i = 0; i < B.size(); i++) {
        auto iter = find(U.begin(), U.end(), B[i]);
        if (iter == U.end()) {
            U.push_back(B[i]);
        }
    }

    // intersection : A * B
    for (int i = 0; i < A.size(); i++) {
        auto iter = find(B.begin(), B.end(), A[i]);
        auto iter2 = find(X.begin(), X.end(), A[i]);
        if (iter != B.end() && iter2 == X.end()) {
            X.push_back(A[i]);
        }
    }
}

```

```

// print A, B
cout << "A : ";
for (int i = 0; i < A.size(); i++) {
    cout << A[i] << " ";
}
cout << endl;
cout << "B : ";
for (int i = 0; i < B.size(); i++) {
    cout << B[i] << " ";
}
cout << endl;

// print vector U
cout << "U = { ";
for (int i = 0; i < U.size(); i++) {
    cout << U[i] << " ";
}
cout << "}" << endl;
cout << "X = { ";
// print vector X
for (int i = 0; i < X.size(); i++) {
    cout << X[i] << " ";
}
cout << "}" << endl;
}

```

3번 문제는, C++에서의 Set STL을 이용하면 더 편하게 풀 수 있을 거라고 생각했지만,

컴알 실습에서 원하는 풀이가 아닐 것이라 생각해 배열(vector)에 담아 풀었다.

교집합이 잘 실행되는지 보기 위해 random하게 숫자를 뽑을 때 1~10까지만 뽑아 봤고,

Time Complexity는 push가 $O(1)$ 이지만, Union과 Interseciton을 만드는 과정에서

$O(n)$ 이 걸리는 find를 n 번씩 수행하기 때문에 $O(n^2)$ 이다.

Space Complexity는 $O(n)$.

```
75     for (int i = 0; i < X.size(); i++) {  
76         cout << X[i] << " ";  
77     }  
78     cout << "}" << endl;  
79 }  
80
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
> cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g test.cpp -o a.out && "/Users/  
jaewonlee/Desktop/PS/"a.out  
6 6  
A : 6 10 4 1 3 4  
B : 2 4 10 8 10 1  
U = { 6 10 4 1 3 2 8 }  
X = { 10 4 1 }
```