

2020105695 김희성

Exercise 1.

- 스택의 동작 방법을 이해하기 위하여 "StackTType.h"에 정의된 StackType클래스를 분석.

Push Pop Top 함수 분석

모든 operator는 initialized 된 후에 사용된다.

- Transformer

Push() : Top이 가르키는 index의 위에 value를 추가한다.

Pop() : Top이 가르키는 index의 값을 삭제한다.

- Observer

Top() : Top이 가르키는 index의 value를 출력한다.

IsFull() : stack이 가득 차있는지 bool return

IsEmpty() : stack에 아무 것도 없는지 bool return

- member variable

top : end index of item. If there is no items, it has -1

`ItemType items[MAX_ITEMS]` : generic data type array, MAX_ITEMS = 5

- 템플릿으로 정의된 스택을 정수형 타입으로 선언하고 스택에 1,2,3,4,5,6을 순서대로 삽입하고 하나씩 꺼내서 출력하는 프로그램을 작성함.

```
int main()
{
    StackType<int> stack1;
    for (int i = 1; i < 7; i++)
        stack1.Push(i);

    for (int i = 1; i < 7; i++)
    {
        cout << stack1.Top() << endl;
        stack1.Pop();
    }
}
```

Exercise 2

스택의 데이터를 변경하지 않고, 기존의 스택과 동일한 값을 가지는 스택을 만드세요

- Default 복사 생성자 사용.

```
13 template<typename T> <T> Provide sample template arguments for IntelliSense -
14 void Copy(const StackType<T>& copied, StackType<T>& copier)
15 {
16     StackType<T> dummy{ copied };
17     StackType<T> inverse_dummy{};
18
19     // to copy stack, make temporal inverse stack
20     while (!dummy.IsEmpty())
21     {
22         inverse_dummy.Push(dummy.Top());
23         dummy.Pop();
24     }
25
26     //
27     while (!inverse_dummy.IsEmpty())
28     {
29         copier.Push(inverse_dummy.Top());
30         inverse_dummy.Pop();
31     }
32 }
33
34
```

Exercise 3

하나의 배열을 이용하여, 두 개의 스택을 구현하는 double stack 클래스 를 작성하세요.

- ❖ 첫 번째 스택은 1000이하의 수를 저장합니다.
 - ❖ 두 번째 스택은 1000을 넘는 수를 저장합니다.
 - ❖ Double stack의 최대 아이템 저장 갯수는 200입니다.
 - ❖ 각 스택의 개수는 정해지지 않았습니다.
- 1000이하의 수로 200개를 저장할 수 있고, 1000이하의 수가 하나도 없을 수도 있습니다.
 - a. 하나의 배열에 stack 2개를 어떻게 구현할 것인가?
 - 작은 것은 array의 앞 부분에 큰 것은 array의 뒷부분에 저장한다.
 - b. A에서 생각한 double stack을 클래스로 정의해 보세요.

```
1 #include <iostream>
2 using namespace std;
3
4 const int MAX_ITEMS = 200;
5
6 class doublestack
7 {
8 private:
9     int top_small; //1000보다 작거나 같은 스택의 top
10    int top_big; // 1000보다 큰 스택의 top
11    int items[MAX_ITEMS];
12 public:
13    doublestack();
14    void Push(int item); //C에서 구현할 push 연산
15    void Print(); //stack 의 상황을 보여줄 수 있는 함수(채점시)
16    bool IsFull()const;
17 };
18
```

- c. double stack 클래스의 멤버 함수 중 Push 연산 부분을 구현해 보세요.

```
void doublestack::Push(int item)
{
    if (IsFull()) //error
        return;

    if (item <= 1000)
        items[++top_small] = item;
    else if (item > 1000)
        items[MAX_ITEMS - 1 - (++top_big)] = item;
}
```

- d. 채점을 위해 저장된 아이템들을 확인 하는 Print()함수를 작성하세요.

❖ 스택 pop 순서로 출력을 해야하며, 1000이하 스택을 출력 후에 1000초과 스택 을 출력하세요.

```
void doublestack::Print()
{
    cout << "small stack" << endl;
    for (int i = top_small; i >= 0; i--)
        cout << " | " << items[i] << " |";
    //const int MAX_ITEMS = 200;
    cout << "\n\n";

    cout << "big stack" << endl;
    for (int i = MAX_ITEMS - 1 - top_big; i <= MAX_ITEMS - 1; i++)
        cout << " | " << items[i] << " |";
    cout << "\n";
}
```

```
small stack
0 | 100 |

big stack
1002 | 1001 | 1001 | 2000 |

C:\Users\user\Desktop\SW\data_structure_algorithm\data_structure_lecture\data_structure\Debug\data_struct
16500) exited with code 0.
Do not automatically close the console when debugging stops, enable Tools->Options->Debugging->Automaticall
e when debugging stops.
Press any key to close this window . . .
```

풀이 과정 :

Pop 이 없어도 private 변수에 접근이 가능해서, indexing을 통해 접근했습니다.

자료구조를 따로 공부하고 있습니다! Array 기반의 자료구조의 장점은 indexing이 편한 것이고, linked list 기반 자료구조는 삽입과 삭제가 편한 것이라고 본 것 같습니다. 한 개씩 빼는 pop보다는 indexing이 좋을 것 같아 위와 같이 작성해봤습니다.

Exercise 4

◆각 스택의 복사본을 다른 것으로 치환하는 함수를 작성하여라. 다음과 같은 사양을 사용하여라(이 함수는 호출 프로그램이다.)

Replace Item

함수 : 모든 oldItem을 newItem으로 바꾼다.

조건 : 스택은 초기화되어 있다.

결과 : 스택에 있는 각각의 oldItem은 newItem으로 바뀌진다.

template로 작성한 stackType이 아닌 int 타입을 사용하는 StackType 클래스를 사용한다. (*경로 : [WWlapplusWWLab,C++3rdWWChapter4WWstackWWStatic](#))

A. ReplaceItem 함수를 StackType 클래스의 클라이언트로 작성한다.

```
void ReplaceItem(StackType& st, int oldItem, int newItem)
{
    StackType dummy{};

    // dummy is inverse stack...-> 차례리 변수 이름을 inverse stack으로 지을것
    while (!st.IsEmpty())
    {
        int top = st.Top();
        if (top == oldItem)
            dummy.Push(newItem);
        else
            dummy.Push(top);
        st.Pop();
    }

    while (!dummy.IsEmpty())
    {
        int top = dummy.Top();
        if (top == oldItem)
            st.Push(newItem);
        else
            st.Push(top);
        dummy.Pop();
    }
}
```

B. ReplaceItem 함수를 StackType의 멤버 함수가 되도록 StackType을 수정한다.

*Template를 사용하지 않는 StackType 클래스를 사용한다.

```
void StackType::ReplaceItem(int old_item, int new_item)
{
    for (int i = 0; i <= top; i++)
    {
        if (items[i] == old_item)
            items[i] = new_item;
    }
}
```