|  |  |
| --- | --- |
|  | **윈도우프로그래밍**  **LAB 04** |
| **분반** | 2 |
| **학번** | 1815060 |
| **이름** | 문정현 |

**# 실습 문제 1**

#1 main.cpp 프로그램 소스(수정한 결과 소스)

#include <iostream>

#include <strstream>

#include "Stack.h"

using namespace std;

char line[100];

void check\_balance(istrstream& is) {

bool flag = true;

char tmp;

int idx = 0;

CStack s1;

while ((tmp = is.get()) != -1) {

cout << tmp;

if ((tmp == '(') || (tmp == '[') || (tmp == '{')) {

s1.push(tmp);

}

else if ((tmp == ')') || (tmp == ']') || (tmp == '}')) {

if (s1.isempty()) { cout << " : MISMATCHING!!" << endl; return; }

else {

switch (tmp) {

case ')':

if (s1.TOP() == '(') { flag = true; }

else { flag = false; }

break;

case ']':

if (s1.TOP() == '[') { flag = true; }

else { flag = false; }

break;

case '}':

if (s1.TOP() == '{') { flag = true; }

else { flag = false; }

break;

}

}

if (flag) { s1.pop(); }

else { cout << " : MISMATCHING!!" << endl; return; }

}

}

if (s1.isempty()) { cout << " : BALANCED!!" << endl; }

else { cout << " : MISMATCHING!!" << endl; }

}

int main() {

char c;

while (cin.get(line, 100)) {

istrstream is(line);

if (cin.eof()) { break; }

check\_balance(is);

cin.get(c);

is.clear();

}

return 0;

}

설명

- 기존 파일에서와 달리 스택을 사용하는 부분에서 사용자가 정의한 스택 클래스에서 스택 객체를 생성한 뒤 클래스 내의 멤버 함수를 실행하였고 이를 이용하여 문장의 mismatch/balance를 확인하게 하였습니다.

- 한 글자씩 출력하면서 열린 괄호를 만난 경우 스택에 해당 글자를 push()하고 닫힌 괄호를 만난 경우에는 스택에 제일 나중에 저장된 값을 읽어서(TOP()을 사용하여 읽음) 서로 짝이 맞는 괄호인 경우는 pop()을 하고 계속해서 남은 글자들을 검사하면서 출력하고 짝이 맞지 않거나 스택이 비어있는 경우 바로 mismatch를 출력하고 함수를 return하여 종료합니다.

- 글자를 다 출력한 경우 스택에 괄호가 남아있으면 mismatch를 출력하고 남아있지 않으면 balanced를 출력하고 return하여 종료합니다.

#2 Stack.cpp 프로그램 소스(수정한 결과 소스)

#include "Stack.h"

#include <iostream>

using namespace std;

CStack::CStack(int size) {

this->capacity = size;

this->top = 0;

if ((this->store = new char[size]) == 0) {

cerr << "Error: can't make CStack" << endl;

}

}

CStack::~CStack() {

delete this->store;

}

// If the stack is full, then push val onto the stack s and return 1,

// otherwise simply return 0 to indicate a failure of the push operation

void CStack::push(char val) {

if (this->top < this->capacity)

this->store[(this->top)++] = val;

else

cerr << "Error: stack full, can't push" << endl;

}

char CStack::pop() {

if (this->top > 0)

return this->store[--(this->top)];

else {

cerr << "Error: pop from an empty stack" << endl;

return 0;

}

}

int CStack::isempty() {

return this->top == 0;

}

char CStack::TOP() {

if (this->top > 0)

return this->store[(this->top)-1];

else {return 0;}

}

설명

- 스택클래스의 생성자에서는 자료들을 저장하기 위한 메모리를 원하는 사이즈만큼 할당받고 스택자료구조를 위한 top 인덱스를 초기화시킵니다. 할당을 못 받은 경우에는 에러 메세지를 출력합니다.

- 스택클래스의 소멸자는 delete를 이용하여 할당받은 메모리를 해제합니다.

- 스택에 자료를 삽입하기 위해 top인덱스가 가르키는 위치에 값을 저장하는 push()를 추가했습니다.

- 스택에 제일 나중에 삽입된 자료를 꺼내기 위해 top인덱스-1 위치에 값을 가져오는 pop()를 추가했습니다.

- 스택이 비어있는지 확인하기 위해 top 인덱스의 위치를 확인하는 isempty()를 추가했습니다.

- 스택에서 제일 나중에 삽입된 자료를 읽기 위해 top인덱스-1위치에 저장된 값을 주는 멤버 함수 TOP()를 추가했습니다.

#3 Stack.h 프로그램 소스(수정한 결과 소스)

#pragma once

class CStack

{

private:

char\* store;

int capacity;

int top;

public:

CStack(int size=100);

~CStack();

char pop();

void push(char val);

int isempty();

char TOP();

};

설명

- 멤버변수는 private으로 멤버함수는 public으로 하여 멤버함수를 통해서만 멤버변수의 접근이 가능하도록 했습니다. CStack의 생성자에서는 디폴트 매개변수로 사이즈 100을 입력해서 생성자 호출시 매개변수가 정보가 없을 경우 size를 100으로 주는 것으로 설정했습니다.

#4 실행 화면

