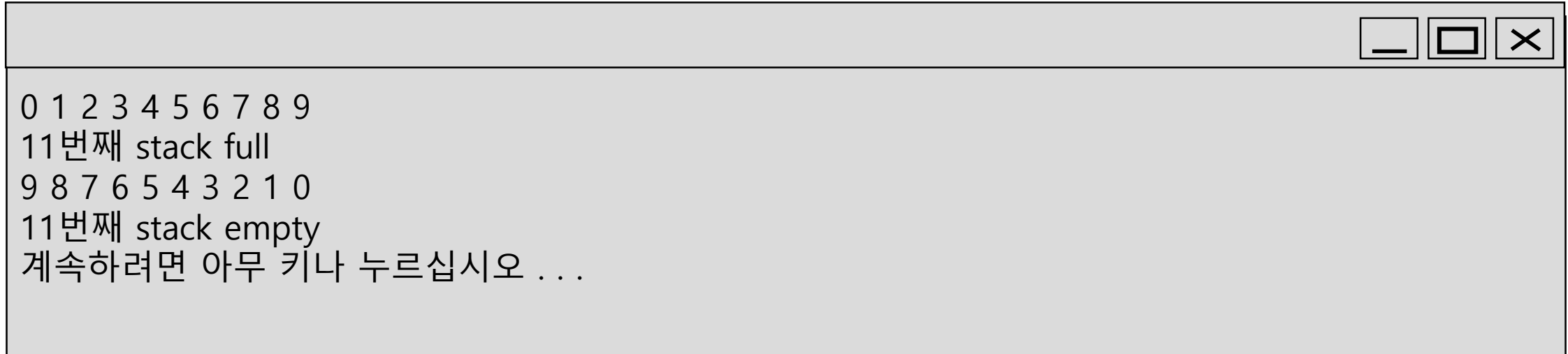


Stack

- 다음과 같이 선언된 정수를 저장하는 스택 클래스 **MyIntStack**을 구현하라.
- **MyIntStack** 스택에 저장 할 수 있는 정수는 최대 10개이다.



```
0 1 2 3 4 5 6 7 8 9
11번째 stack full
9 8 7 6 5 4 3 2 1 0
11번째 stack empty
계속하려면 아무 키나 누르십시오 ...
```

Stack

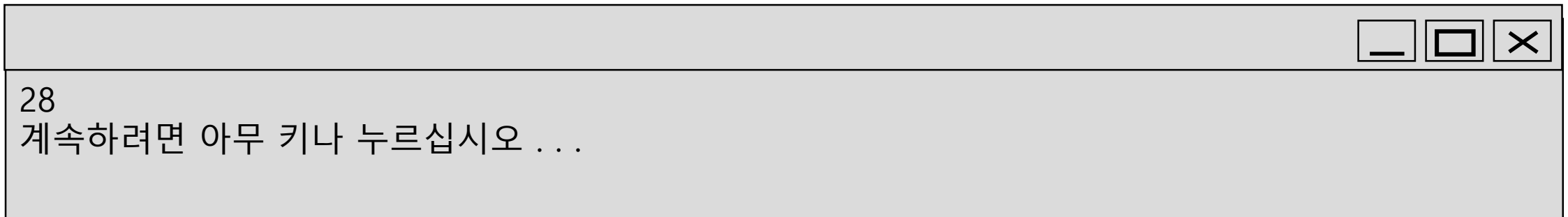
```
class MyIntStack {  
    int p[10]; // 최대 10개의 정수 저장  
    int tos; // 스택의 꼭대기를 가리키는 인덱스  
public:  
    MyIntStack( );  
    bool push(int n); // 정수 n 푸시. 꽉 차 있으면 false, 아니면 true 리턴  
    bool pop(int &n); // 팝하여 n에 저장. 스택이 비어 있으면 false, 아니면 true 리턴  
}
```

Stack

```
int main( ){
    MyIntStack a;
    for(int i=0; i<11; i++){ //11개를 푸시하면, 마지막에는 stack full이 된다.
        if(a.push(i)) cout << i << ' '; //푸시된 값 에코
        else cout << endl << i+1 << "번째 stack full" << endl;
    }
    int n;
    for(int i=0; i<11; i++){ //11개를 팝하면, 마지막에는 stack empty가 된다.
        if(a.pop(n)) cout << n << ' '; // 팝한 값 출력
        else cout << endl << i+1 << "번째 stack empty";
    }
    cout << endl;
}
```

Accumulator 클래스

- 클래스 **Accumulator**는 **add()** 함수를 통해 계속 값을 누적하는 클래스로서, 다음과 같이 선언된다. **Accumulator** 클래스를 구현하라.



Accumulator 클래스

```
class Accumulator {  
    int value;  
public:  
    Accumulator(int value); // 매개변수 value로 멤버 value를 초기화한다.  
    Accumulator& add(int n); // value에 n을 더해 값을 누적한다.  
    int get( ); // 누적된 값 value를 리턴한다.  
}
```

```
int main( ){  
    Accumulator acc(10);  
    acc.add(5).add(6).add(7); // acc의 value 멤버가 28이 된다.  
    cout << acc.get( ); // 28 출력  
}
```

find 함수 작성

- find() 함수의 원형은 다음과 같다. 문자열 a에서 문자 c를 찾아, 문자 c가 있는 공간에 대한 참조를 리턴 한다. 만일 문자 c를 찾을 수 없다면 success 참조 매개 변수에 false를 설정한다. 물론 찾게 되면 success에 true를 설정한다.

```
char& find(char a[], char c, bool& success);
```

```
int main(){
    char s[] = "Mike";
    bool b = false;
    char& loc = find(s, 'M', b);
    if(b == false){
        cout << "M을 발견할 수 없다" << endl;
        return 0;
    }
    loc = 'm';
    cout << s << endl;
}
```

increaseBy ()

- 다음 Circle 클래스를 main문과 increaseBy() 함수의 목적에 맞게 실행 되도록 수정하여라.

```
class Circle {  
    int radius;  
public:  
    Circle(int r) { radius = r; }  
    int getRadius() { return radius; }  
    void setRadius(int r) { radius = r; }  
    void show() { cout << "반지름이 " << radius << "인 원 " << endl; };  
  
    //Circle 객체 b를 a에 더하여 a를 키우고자 다음 함수를 작성하였다.  
    void increaseBy(Circle a, Circle b){  
        int r = a.getRadius() + b.getRadius();  
        a.setRadius(r);  
    }  
  
int main(){  
    Circle x(10), y(5);  
    increaseBy(x,y); // x에 반지름이 15인 원을 만들고자 한다.  
    x.show();  
}
```