객체지향프로그래밍  
HW08 MyStack

C211123 이준선

2023. 5. 12.

# 코드

// SPDX-FileCopyrightText: © 2023 Lee Jun Seon <limazero14@gmail.com>

// SPDX-License-Identifier: Apache-2.0

#include <stdexcept>

#include <string>

#include <cstring>

#include <iostream>

class BaseArray

{

private:

int capacity;

int \*mem;

protected:

explicit BaseArray(int capacity = 100);

virtual ~BaseArray();

void put(int index, int val);

int get(int index);

[[nodiscard]] int getCapacity() const;

void putCapacity(int newCap);

int \*getMem();

void putMem(int \*start);

void deleteMem();

};

BaseArray::BaseArray(int capacity)

{

this->capacity = capacity;

this->mem = new int[static\_cast<unsigned long>(capacity)];

}

BaseArray::~BaseArray()

{

delete[] mem;

}

void BaseArray::put(int index, int val)

{

if (index < 0 || index >= capacity)

{

const auto err\_msg = "Out of range\nExpected range: 0 ~ " + std::to\_string(capacity - 1) +

"\nActual index: " + std::to\_string(index);

throw std::out\_of\_range(err\_msg);

}

mem[index] = val;

}

int BaseArray::get(int index)

{

if (index < 0 || index >= capacity)

{

const auto err\_msg = "Out of range\nExpected range: 0 ~ " + std::to\_string(capacity - 1) +

"\nActual index: " + std::to\_string(index);

throw std::out\_of\_range(err\_msg);

}

return mem[index];

}

int BaseArray::getCapacity() const

{

return capacity;

}

void BaseArray::putCapacity(int newCap)

{

capacity = newCap;

}

int \*BaseArray::getMem()

{

return mem;

}

void BaseArray::putMem(int \*start)

{

mem = start;

}

void BaseArray::deleteMem()

{

delete[] mem;

}

class MyStack : public BaseArray

{

public:

explicit MyStack(int capacity = 100);

~MyStack() override = default;

void push(int n);

int pop();

[[nodiscard]] int capacity() const;

[[nodiscard]] int length() const;

private:

int tos; /// top of stack [index], initial value = -1

};

MyStack::MyStack(int capacity) : tos(-1)

{

if (capacity <= 0)

{

throw std::invalid\_argument("Capacity must be positive");

}

putCapacity(capacity);

putMem(new int[static\_cast<unsigned long>(capacity)]);

}

void MyStack::push(const int n)

{

// If the stack is full

if (length() == getCapacity())

{

int new\_capacity = getCapacity() \* 2;

int \*new\_int\_arr = new int[static\_cast<unsigned long>(new\_capacity)];

memcpy(new\_int\_arr, getMem(), static\_cast<unsigned long>(getCapacity()) \* sizeof(int));

putCapacity(new\_capacity);

deleteMem();

putMem(new\_int\_arr);

}

put(++tos, n);

}

int MyStack::pop()

{

if (tos < 0)

{

throw std::underflow\_error("Stack is empty");

}

int retVal = get(tos--);

// Reduce memory size when the stack is only 1/4 full

if (tos + 1 <= getCapacity() / 4 && getCapacity() > 2)

{

int newCapacity = getCapacity() / 2;

int \*newMem = new int[static\_cast<unsigned long>(newCapacity)];

memcpy(newMem, getMem(), static\_cast<unsigned long>(newCapacity) \* sizeof(int));

putCapacity(newCapacity);

deleteMem();

putMem(newMem);

}

return retVal;

}

int MyStack::capacity() const

{

return getCapacity();

}

int MyStack::length() const

{

return tos + 1;

}

int main()

{

using namespace std;

MyStack mStack(1);

int n;

cout << "스택에 삽입할 5개의 정수를 입력하라>> ";

for (int i = 0; i < 5; i++)

{

cin >> n;

mStack.push(n); // 스택에 푸시

}

cout << "스택 용량:" << mStack.capacity() << ", 스택 크기:" << mStack.length() << endl;

cout << "스택의 모든 원소를 팝하여 출력한다>> ";

while (mStack.length() > 0)

{

int k = mStack.pop();

cout << k << ' '; // 스택에서 팝

}

cout << "\n스택 용량:" << mStack.capacity() << ", 스택 크기:" << mStack.length() << endl;

}

# 실행 결과

