Unreal second :

assignment 3

17조 김영빈

목차

개요

- 템플릿과 클래스를 이용해 C++ 에서 vector class 의 기능을 몇가지 구현하는 과제이다.

- Push_back, pop_back, size, capacity 기능을 필수 구현한다.

- Resize, sort 기능을 추가 구현한다.

클래스

main.cpp simpleVector.h

```
1   #include <algorithm>
2   #include <iostream>
3
4   using namespace std;
5
6   template <typename Element> class SimpleVector
7   {
8       private:
9           Element** s_vec;
10          int cnt;
11          int space;
```

- 템플릿을 사용해 객체를 선언하면서 자료형을 결정한다.
- 이는 기존 vector 를 선언할 때 Vector< type > name 형태를 고려해 자료형에 의존하지 않고 데이터를 받을 수 있게 하는 것이다.
- 템플릿 포인터를 원소로 가진 동적 배열을 멤버 변수로 가진다.
- Cnt 는 원소의 개수, space 는 vector크기이다.

생성자

```
13      public:
14          SimpleVector(int input_idx = 10)
15          {
16              space=input_idx;
17              s_vec = new Element*[space];
18              cnt=0;
19              int idx=0;
20              while(idx<space)
21              {
22                  s_vec[idx]=nullptr;
23                  idx++;
24              }
25          }
26
```

기본 생성자

- 생성자에 필요한 정보는 vector 의 크기 하나이다.
- 기본 생성자로 크기가 10 인 vector 를 생성한다.
  매개변수를 받는 경우 그 값의 크기만큼 생성한다.
- Vector 내부를 nullptr 로 초기화 한다.

생성자

```
27      SimpleVector(const SimpleVector& obj)
28      {
29          cnt=obj.cnt;
30          space=obj.space;
31          s_vec = new Element*[space];
32          int idx=0;
33          while(idx<space)
34          {
35              s_vec[idx]=obj.s_vec[idx];
36              idx++;
37          }
38      }
39
40      SimpleVector& operator=(const SimpleVector& obj)
41      {
42          if(this != &obj)
43          {
44              delete[] s_vec;
45
46              space = obj.space;
47              cnt = obj.cnt;
48              s_vec = new Element*[space];
49              int idx=0;
50              while(idx<space)
51              {
52                  s_vec[idx]=obj.s_vec[idx];
53                  idx++;
54              }
55          }
56          return *this;
57      }
58
```

복사 생성자

- 복사할 vector 의 원소 개수, 크기를 먼저 복사하고 그 값만큼 복사를 반복한다.
- 오버로드를 통해 깊은 복사가 가능하게 구현했다.
- 자기 자신이 아니라면 복사할 vector 를 참조하고 복사한다.

생성자

```
59          ~SimpleVector()
60          {
61              delete[] s_vec;
62              cnt=0;
63              space = 0;
64              cout<<"destructor"<<endl;
65          }
```

소멸자

- 프로그램이 끝나는 시점에 호출된다.
- 할당된 메모리를 해제한다.
- 확인을 위해 결과를 출력하도록 했다.

Push_back( )

```
void push_back(Element* val)
{
    if(cnt<space)
    {
        s_vec[cnt]=val;
    }
    else
    {
        space+=5;
        Element** after = new Element*[space];
        int idx=0;
        while(idx<space)
        {
            if(idx<cnt)
            {
                after[idx]=s_vec[idx];
            }
            else
            {
                after[idx]=nullptr;
            }
            idx++;
        }
        delete[] s_vec;
        after[cnt]=val;
        idx=0;
        s_vec=new Element*[space];
        while(idx<space)
        {
            if(idx<cnt+1)
            {
                s_vec[idx]=after[idx];
            }
            else
            {
                s_vec[idx]=nullptr;
            }
            idx++;
        }
        delete[] after;
    }
    cnt++;
}
```

- 마지막 원소 뒤에 새로운 원소를 삽입한다.
- Vector 에 빈자리가 없는 경우 기존 크기보다 5만큼 큰 vector를 새로 선언한다.
- 새 vector 에 기존 값을 복사하고 메모리를 해제한다.
- 변경된 cnt, space 를 반영한다.
- 기본 배열에 크기 조정하는 기능이 없어서 이렇게 구현했다.

Pop_back( )

```
111         void pop_back()
112         {
113             int idx=0;
114             if(cnt>0)
115             {
116                 cout<<"pop is work?"<<endl;
117                 s_vec[cnt-1]=nullptr;
118                 cnt--;
119             }
120         }
121
```

- Class 내부의 cnt 값으로 마지막 원소가 몇 번째 인지 알고있다.
- Vector 의 cnt index 원소를 nullptr 로 바꿔 구현했다.

Size( )

```
int size()
{
    return cnt;
}
```

- Cnt 를 반환해 원소 개수를 반환한다.

Capacity( )

```
int capacity()
{
    return space;
}
```

- Class 내부 space 값으로 vector 크기를 반환한다.

SortData( )

```
void sortData()
{
    sort(s_vec, s_vec+cnt, [](Element* a, Element* b)
    {
        if (a == nullptr && b == nullptr) return false;
        if (a == nullptr) return false;
        if (b == nullptr) return true;
        return *a < *b;
    });
}
```

- Vector 원소는 포인터이다.
- 실제 가리키는 값을 기준으로 정렬하기 위해 이렇게 구현했다.

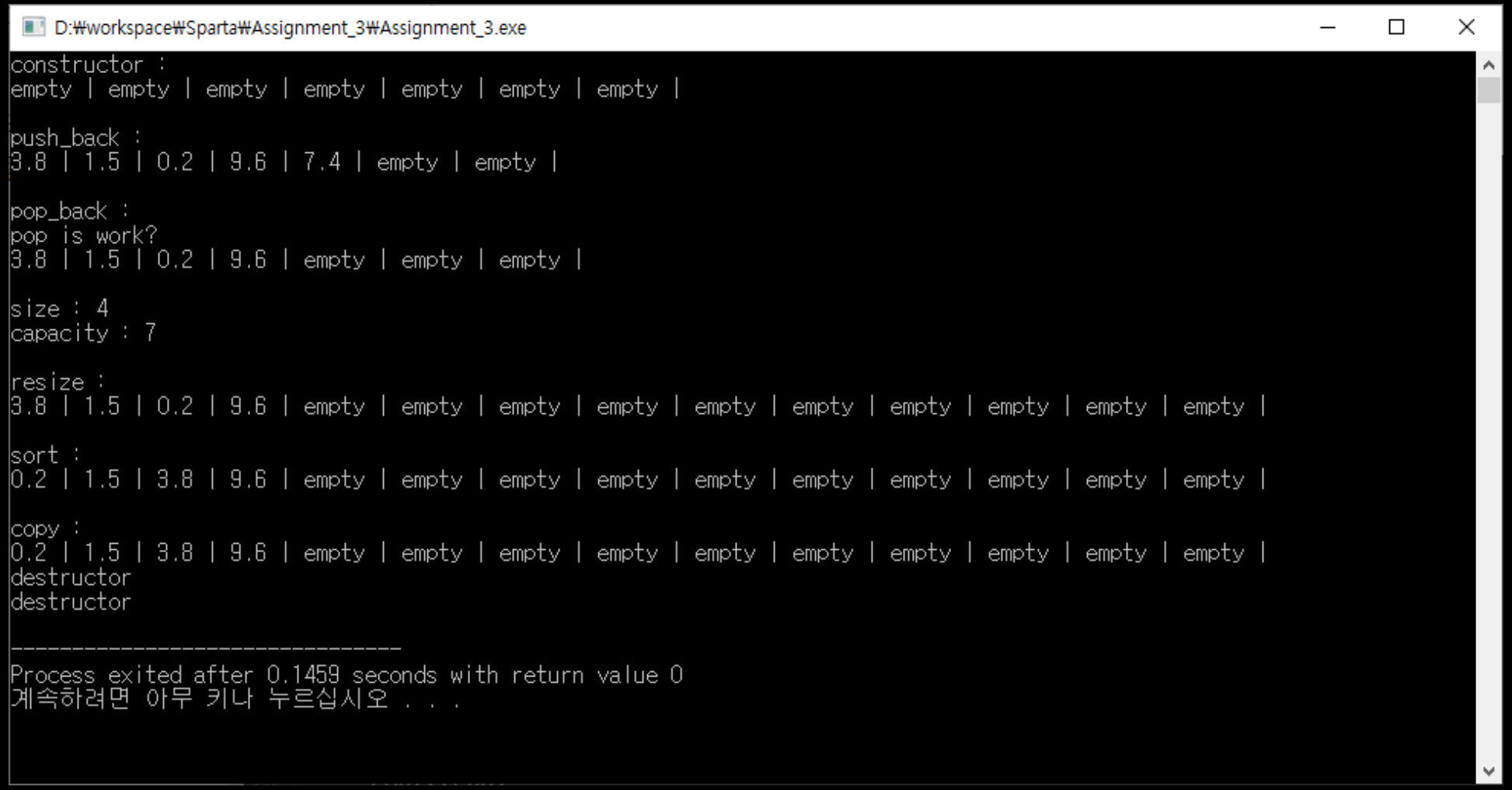
Resize( )

```
void resize(int newCapacity)
{
    if(newCapacity > space)
    {
        space = newCapacity;
        Element** after = new Element*[newCapacity];
        int idx=0;

        while(idx < space)
        {
            if(idx<cnt)
            {
                after[idx]=s_vec[idx];
            }
            else
            {
                after[idx]=nullptr;
            }
            idx++;
        }
        delete[] s_vec;
        idx=0;
        s_vec = new Element*[newCapacity];
        while(idx < space)
        {
            if(idx<cnt)
            {
                s_vec[idx]=after[idx];
            }
            else
            {
                s_vec[idx]=nullptr;
            }
            idx++;
        }
        delete[] after;
    }
}
```

- 배열은 크기 조정 기능이 없어 이렇게 구현했다.
- 새로운 동적 배열을 선언하고 기존 값을 복사한다.
- 기존 배열 메모리를 해제한다.

결과


D:₩workspace₩Sparta₩Assignment_3₩Assignment_3.exe
constructor :
empty | empty | empty | empty | empty | empty | empty |
push_back :
3.8 | 1.5 | 0.2 | 9.6 | 7.4 | empty | empty |
pop_back :
pop is work?
3.8 | 1.5 | 0.2 | 9.6 | empty | empty | empty |
size : 4
capacity : 7
resize :
3.8 | 1.5 | 0.2 | 9.6 | empty | empty | empty | empty | empty | empty | empty | empty | empty | empty |
sort :
0.2 | 1.5 | 3.8 | 9.6 | empty | empty | empty | empty | empty | empty | empty | empty | empty | empty |
copy :
0.2 | 1.5 | 3.8 | 9.6 | empty | empty | empty | empty | empty | empty | empty | empty | empty | empty |
destructor
destructor

Process exited after 0.1459 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .