

Data Structures 2024



Homework #1

- (i) Make class *MyStringVector* similar to class *vector<string>* in STL, and also (ii) write a test program to check that all the member functions/operators of your class *MyStringVector* work correctly.

```
class MyStringVector {  
    public:  
        ...  
    private:  
        string *str_data;  
        ...  
};
```

Note that class *string* is defined in the library *<string>*.

Constraints

- Use a “pointer to a string” variable and dynamic memory allocation by the `new` operator.
 - `string *str_data; // private member`
- Do not use any static array!

Members to be Implemented

- Default constructor
 - `MyStringVector();`
- Copy constructor for deep copy
 - `MyStringVector(const MyStringVector& v);`
- Destructor
 - `~MyStringVector();`
- Assignment operator (=) for deep copy
 - Chaining assignment should be possible.

pre: 메모리 할당
post: str-data = nullptr
capacity = 0, size = 0
→ 유효한 vector 생성

pre: other 가 valid 한 상태, 복사가능
post: this 의 private 변수들에
str-data, size, capacity 복사

pre: constructor가 호출된적 있음,
post: str-data 에 할당된 메모리가 있다면 해제

pre: ① other 는 valid
② 자기함당 x ← (this == &other 이면 return *this;)
③ 메모리 할당 예외 x
post: ① this 는 other 의 size, capacity 복사
② 이전 this 메모리 해제
③ 자기함당 호출시 자신 반환

Members to be Implemented

- Operator: +=

- Appends RHS object to LHS one.

pre: ① other 는 valid
② this 의 capacity는 충분해야한다.
③ ④가 아니라면 할당이 가능해야한다.
④ other. getSize > 0 이면 this 공간 할당 / 복사는
여기 여의 간다.
⑤ &other != this

post: ① this 의 size ± other 의 size.
② other 의 모든 요소 this 의 뒤에 추가
③ this 객체의 capacity가 필요에 의해 확장
④ this 이 원 객체, other 변형
⑤ 메모리 할당 필요시 할당

- Operator: []

- Returns a reference to the string element at the requested position in the vector container.
- Bound-Check: if the requested position is out of range, it should output some messages and terminate the program.

pre: index < size

post: index 번째 요소에 대한 const 참조 반환

Members to be Implemented

pre: vector is not empty
post: (마지막 요소 제거) + (size -- 1)

- `void pop_back();`
 - Removes the last string element in the vector, effectively reducing the vector size by one and invalidating all references to it.
- `void push_back(string s);`
 - Adds a new string element at the end of the vector, after its current last string element. The content of this new string element is initialized to a copy of s.
- `size_t capacity() const;`
 - Returns the size of the allocated storage space for the elements of the vector container.

pre: Vector 가 valid.

post: size vector의 마지막에 영감을
size += 1

reserve() 호출로 capacity += 2 인가 있음.

pre: capacity 가 초과되어 있음.

this 가 valid.

post: capacity 를 return. this 는 변함 x

Members to be Implemented

- `size_t size() const;`

- Returns the number of string elements in the vector container.

pre: this가 valid.
size가 증가할 되어 있어야 한다
post: size 반환
this는 변화 X

- `void shrink_to_fit();`

- Requests the container to reduce its capacity to fit its size.

pre: ① this가 valid.
② data가 유효한 메모리 영역을 가리킴.
③ capacity ≥ size
post: ① capacity == size
② (size == 0) → delete [] str-data; capacity = 0;
③ size는 변화 X

- `void reserve(size_t n);`

- Requests that the capacity of the allocated storage space for the string elements of the vector container be at least enough to hold n string elements.

Note that `size_t` is defined in the library `<cstdlib>`.

pre: new-capacity ≥ capacity
this가 valid.
(data == nullptr) → 빈 벡터
post: capacity가 new-capacity로 증가.
새 메모리 블록 할당. 기존 데이터 복사.
size 변화 X

Members to be Implemented

pre: this 가 valid.

post: vector 의 방 여부에 따라 true: false.
vector 변경

- `bool is_empty() const;`

- Returns whether the vector container is empty, i.e., whether its size is 0.

pre: this 가 valid. data 와 size 는 valid.

post: size = 0.

데이터는 유지.

capacity 유지. 2 가 2 배를 가짐.

- `void clear();`

- All the string elements of the vector are dropped: they are removed from the vector container, leaving the container with a size of 0 and a default capacity.

Due Date

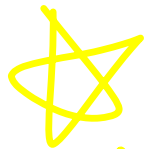
- Soft deadline: **Oct. 12, 2024**
- Hard deadline: Oct. 18, 2024

Submission date	Deduction rate
Oct. 13	10 %
Oct. 14	20 %
Oct. 15	30 %
Oct. 16 – Oct. 18	50 %

Notice

- Do not use any container class in STL, such as “vector”! `#include <vector>` ~~XX~~
- Do not use “printf()” and “scanf()” functions!
- You should never use global variables `C func()` ~~XX~~
- Each member function/operator should have its precondition and postcondition as comments

– E.g.,



return-type *MyStringVector::memberFunction(...);*



// precondition: ...



// postcondition: ...

Notice (cont'd)

- Your class will be tested in another test program.
- You should submit a compressed file (**HW1_your-ID.zip**) containing the following four files to the website (<https://klas.kw.ac.kr/>)
 - **HW1_your-ID.hwp/ docx/ pdf** // report document
 - **HW1_your-ID.cpp/.cc** // your main function (a test program)
 - **MyStringVector.cpp/.cc** // class implementation only
 - **MyStringVector.h** // class documentation & definition only

Notice (cont'd)

- Source code
 - It should be compiled in **Visual Studio 2010 or higher, or g++**
 - **You should note your environment in your report.**
 - Your name and student ID should be noted at the top of your source code in the form of comment
- Report
 - Free format
 - But, it must include several examples of your program and your own discussion
 - It will be an important factor for getting a good score