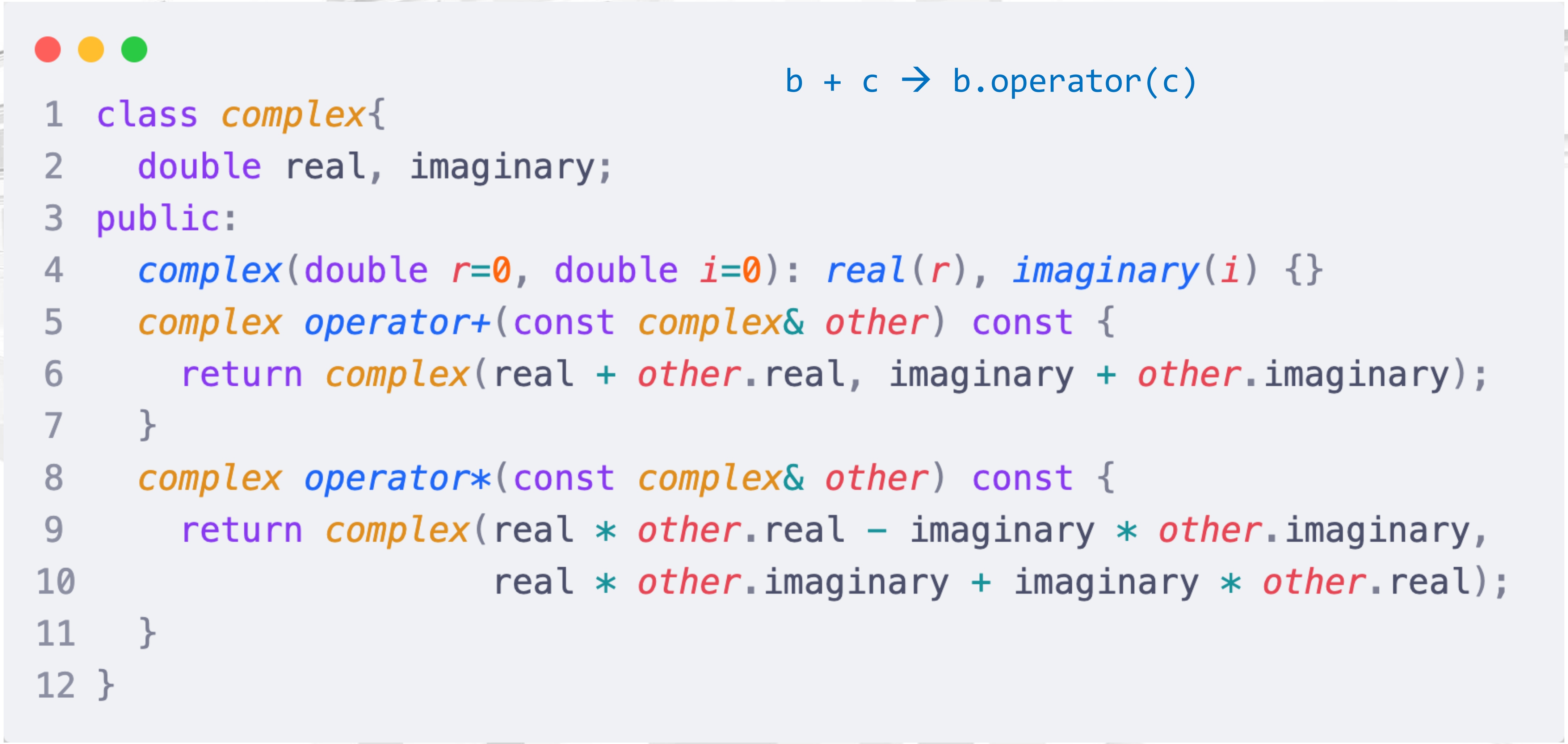
Pervasive Elastic MetaLearning Laboratory
Department of Computer Engineering
Hongik University

자료구조 실습 3 - 추상자료형(ADT) Bag 구현하기

2025 DS ex3

연산자 오버로딩

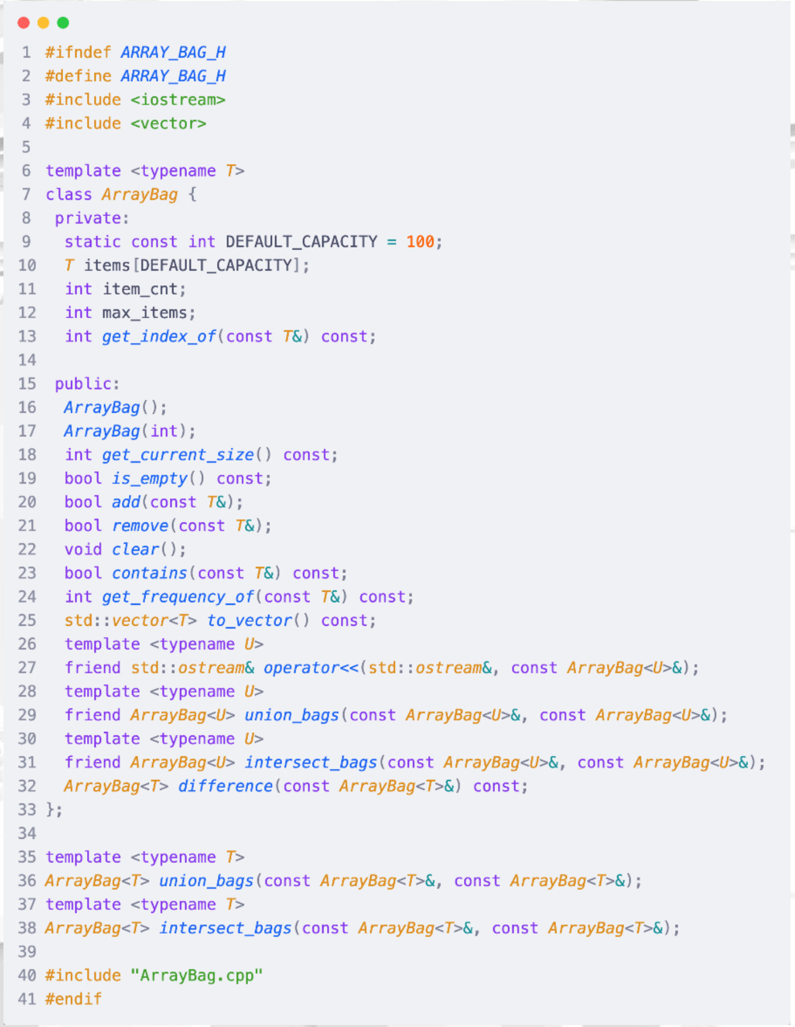
- 간명함을 위함
  - 예시
    - x + y * z
    - multiply y by z and and add the result to x
- 기본 자료형에 대한 연산들은 정의되어 있으나 사용자 정의 타입은 그렇지 않으므로, 편리를 제공하기 위함.
- 형태 : (반환형) operator(연산자) (연산자가 받는 인자)



```cpp
class complex{
  double real, imaginary;
public:
  complex(double r=0, double i=0): real(r), imaginary(i) {}
  complex operator+(const complex& other) const {
    return complex(real + other.real, imaginary + other.imaginary);
  }
  complex operator*(const complex& other) const {
    return complex(real * other.real - imaginary * other.imaginary,
                   real * other.imaginary + imaginary * other.real);
  }
}
```

b + c → b.operator(c)

ArrayBag

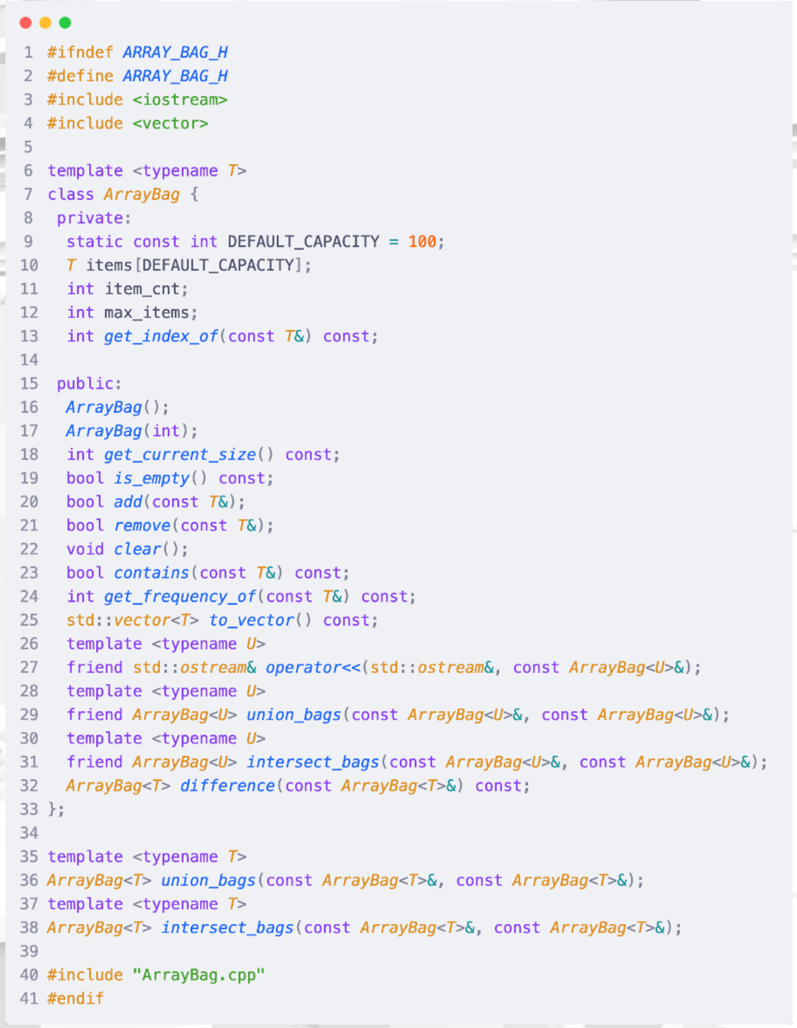


```cpp
#ifndef ARRAY_BAG_H
#define ARRAY_BAG_H
#include <iostream>
#include <vector>

template <typename T>
class ArrayBag {
 private:
  static const int DEFAULT_CAPACITY = 100;
  T items[DEFAULT_CAPACITY];
  int item_cnt;
  int max_items;
  int get_index_of(const T&) const;

 public:
  ArrayBag();
  ArrayBag(int);
  int get_current_size() const;
  bool is_empty() const;
  bool add(const T&);
  bool remove(const T&);
  void clear();
  bool contains(const T&) const;
  int get_frequency_of(const T&) const;
  std::vector<T> to_vector() const;
  template <typename U>
  friend std::ostream& operator<<(std::ostream&, const ArrayBag<U>&);
  template <typename U>
  friend ArrayBag<U> union_bags(const ArrayBag<U>&, const ArrayBag<U>&);
  template <typename U>
  friend ArrayBag<U> intersect_bags(const ArrayBag<U>&, const ArrayBag<U>&);
  ArrayBag<T> difference(const ArrayBag<T>&) const;
};

template <typename T>
ArrayBag<T> union_bags(const ArrayBag<T>&, const ArrayBag<T>&);
template <typename T>
ArrayBag<T> intersect_bags(const ArrayBag<T>&, const ArrayBag<T>&);

#include "ArrayBag.cpp"
#endif
```

friend

- 은닉화 무시
- private, protect 멤버에 접근 가능

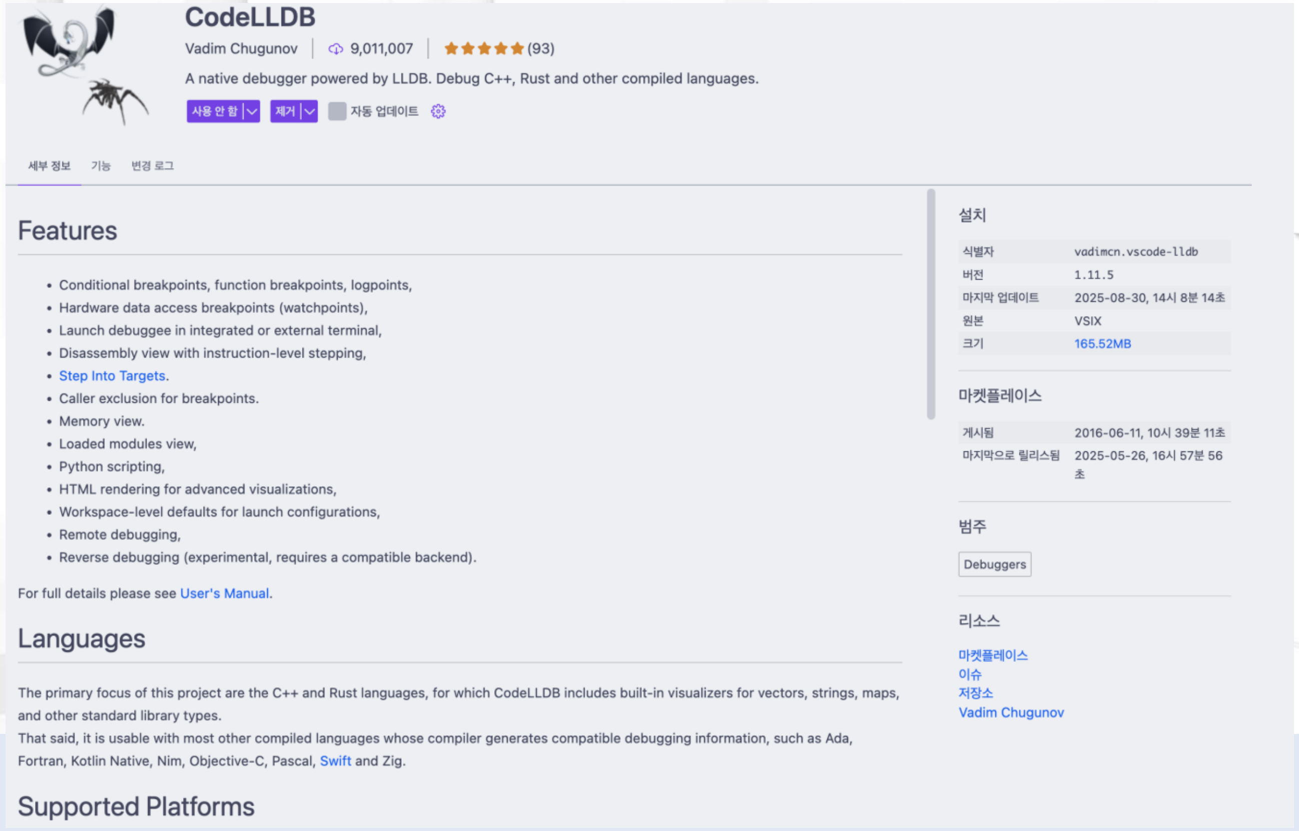
ArrayBag



```cpp
#ifndef ARRAY_BAG_H
#define ARRAY_BAG_H
#include <iostream>
#include <vector>

template <typename T>
class ArrayBag {
 private:
  static const int DEFAULT_CAPACITY = 100;
  T items[DEFAULT_CAPACITY];
  int item_cnt;
  int max_items;
  int get_index_of(const T&) const;

 public:
  ArrayBag();
  ArrayBag(int);
  int get_current_size() const;
  bool is_empty() const;
  bool add(const T&);
  bool remove(const T&);
  void clear();
  bool contains(const T&) const;
  int get_frequency_of(const T&) const;
  std::vector<T> to_vector() const;
  template <typename U>
  friend std::ostream& operator<<(std::ostream&, const ArrayBag<U>&);
  template <typename U>
  friend ArrayBag<U> union_bags(const ArrayBag<U>&, const ArrayBag<U>&);
  template <typename U>
  friend ArrayBag<U> intersect_bags(const ArrayBag<U>&, const ArrayBag<U>&);
  ArrayBag<T> difference(const ArrayBag<T>&) const;
};

template <typename T>
ArrayBag<T> union_bags(const ArrayBag<T>&, const ArrayBag<T>&);
template <typename T>
ArrayBag<T> intersect_bags(const ArrayBag<T>&, const ArrayBag<T>&);

#include "ArrayBag.cpp"
#endif
```

VSCode 설정

- 표준 입출력 리다이렉션
  - 직접 타이핑하는 대신에 파일로 표준 입력(c++; cin), 표준 출력(c++;cout)을 받는 기능
- LLDB 설치(1주차 참고) + CodeLLDB 설치(표준 입출력 지원)



CodeLLDB

Vadim Chugunov | 9,011,007 | ★★★★★ (93)

A native debugger powered by LLDB. Debug C++, Rust and other compiled languages.

Features

- Conditional breakpoints, function breakpoints, logpoints,
- Hardware data access breakpoints (watchpoints),
- Launch debuggee in integrated or external terminal,
- Disassembly view with instruction-level stepping,
- Step Into Targets.
- Caller exclusion for breakpoints.
- Memory view.
- Loaded modules view,
- Python scripting,
- HTML rendering for advanced visualizations,
- Workspace-level defaults for launch configurations,
- Remote debugging,
- Reverse debugging (experimental, requires a compatible backend).

For full details please see User's Manual.

Languages

The primary focus of this project are the C++ and Rust languages, for which CodeLLDB includes built-in visualizers for vectors, strings, maps, and other standard library types.

That said, it is usable with most other compiled languages whose compiler generates compatible debugging information, such as Ada, Fortran, Kotlin Native, Nim, Objective-C, Pascal, Swift and Zig.

Supported Platforms

설치

식별자	vadimcn.vscode-lldb
버전	1.11.5
마지막 업데이트	2025-08-30, 14시 8분 14초
원본	VSIX
크기	165.52MB

마켓플레이스

게시됨	2016-06-11, 10시 39분 11초
마지막으로 릴리스됨	2025-05-26, 16시 57분 56초

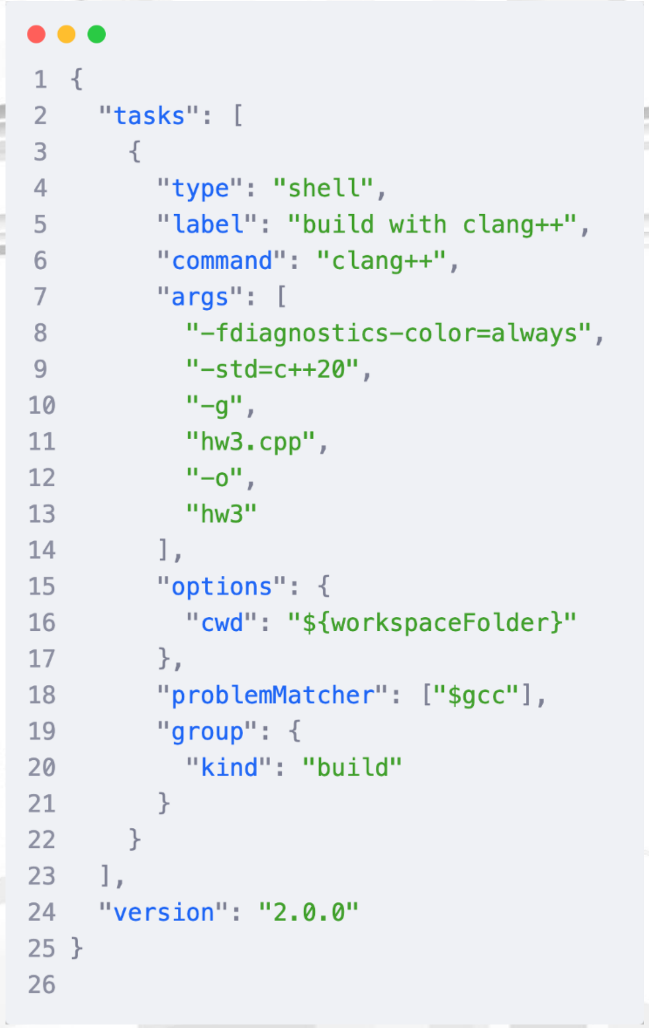
범주

Debuggers

리소스

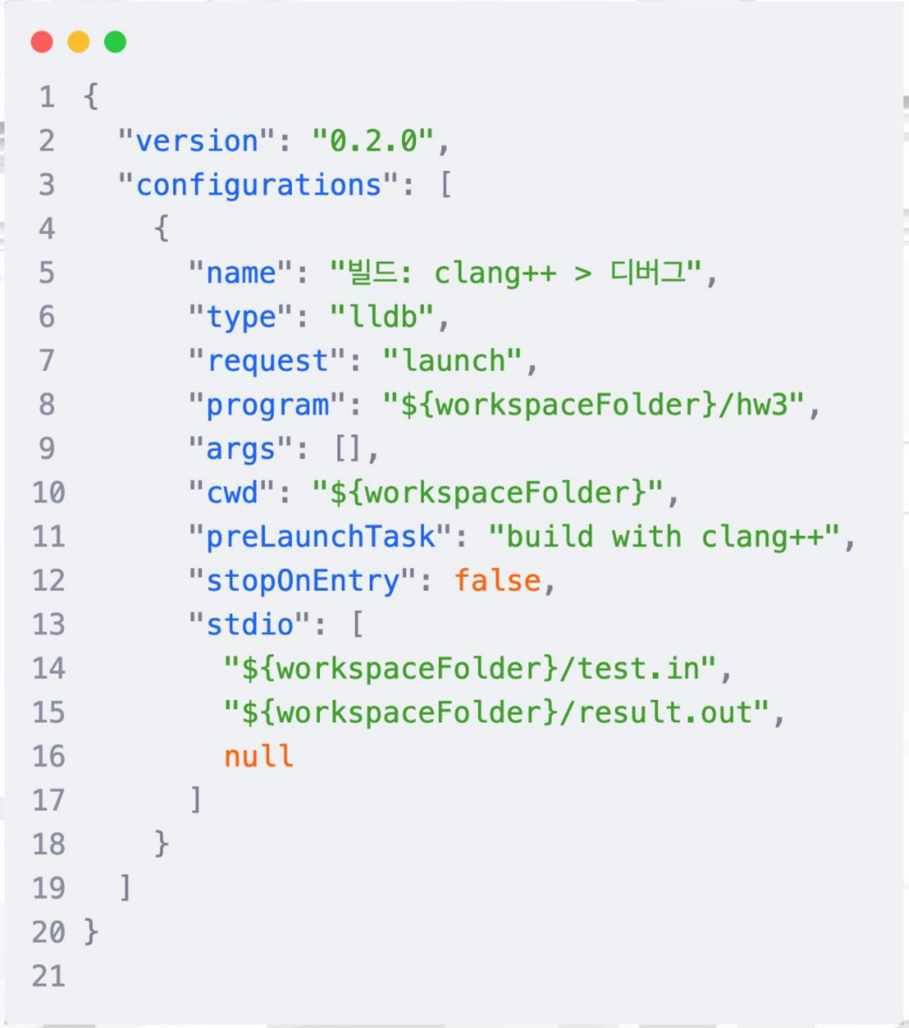
마켓플레이스
이슈
저장소
Vadim Chugunov

task.json



```json
{
  "tasks": [
    {
      "type": "shell",
      "label": "build with clang++",
      "command": "clang++",
      "args": [
        "-fdiagnostics-color=always",
        "-std=c++20",
        "-g",
        "hw3.cpp",
        "-o",
        "hw3"
      ],
      "options": {
        "cwd": "${workspaceFolder}"
      },
      "problemMatcher": ["$gcc"],
      "group": {
        "kind": "build"
      }
    }
  ],
  "version": "2.0.0"
}
```

launch.json

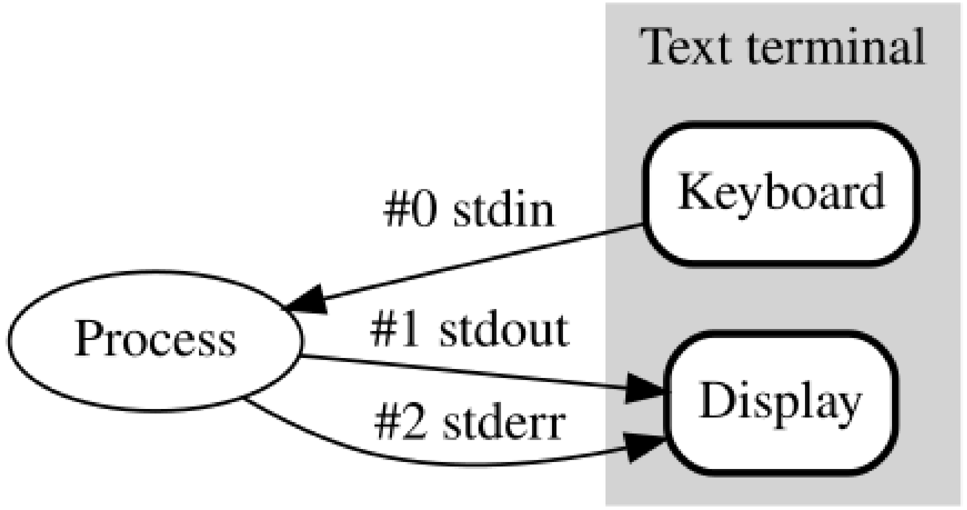


```json
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "빌드: clang++ > 디버그",
      "type": "lldb",
      "request": "launch",
      "program": "${workspaceFolder}/hw3",
      "args": [],
      "cwd": "${workspaceFolder}",
      "preLaunchTask": "build with clang++",
      "stopOnEntry": false,
      "stdio": [
        "${workspaceFolder}/test.in",
        "${workspaceFolder}/result.out",
        null
      ]
    }
  ]
}

```

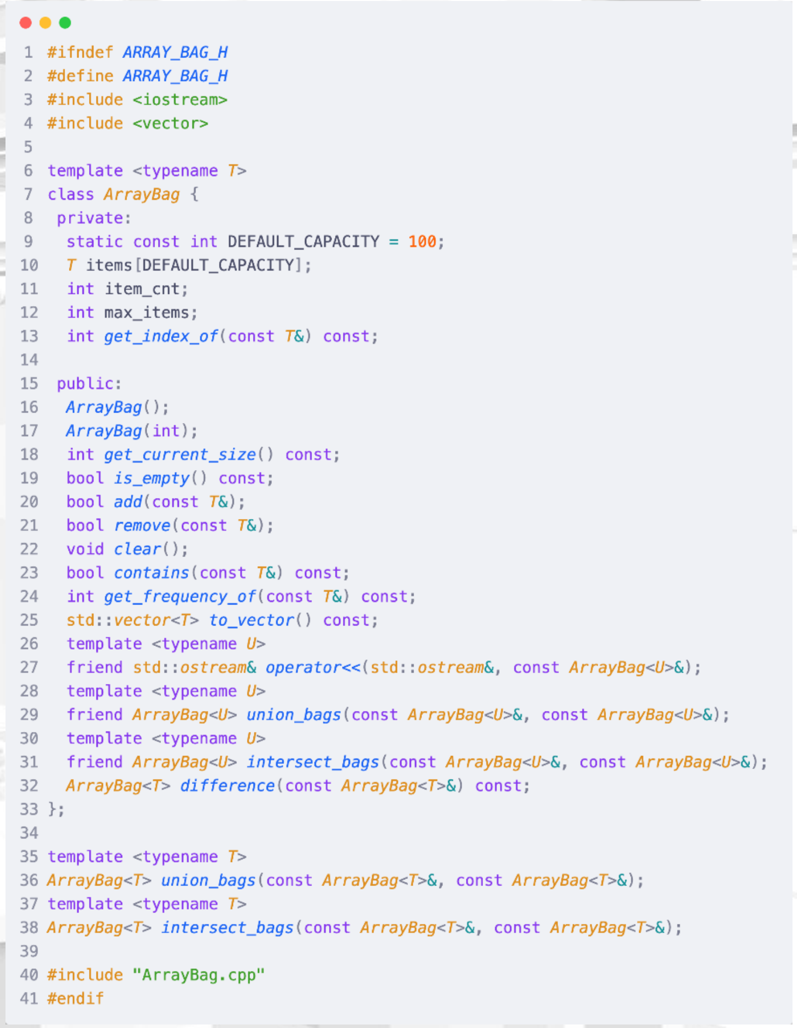
Stdio Redirection

- 프로그램의 입/출력 파일이나 다른 스트림으로 전달할 때 사용
- C++ 기준으로는 cin, cout 등과 관련 있음

- 파이프(|)는 결과를 다른 프로그램으로 넘겨줄 때 사용된다는 점에서 차이가 있음

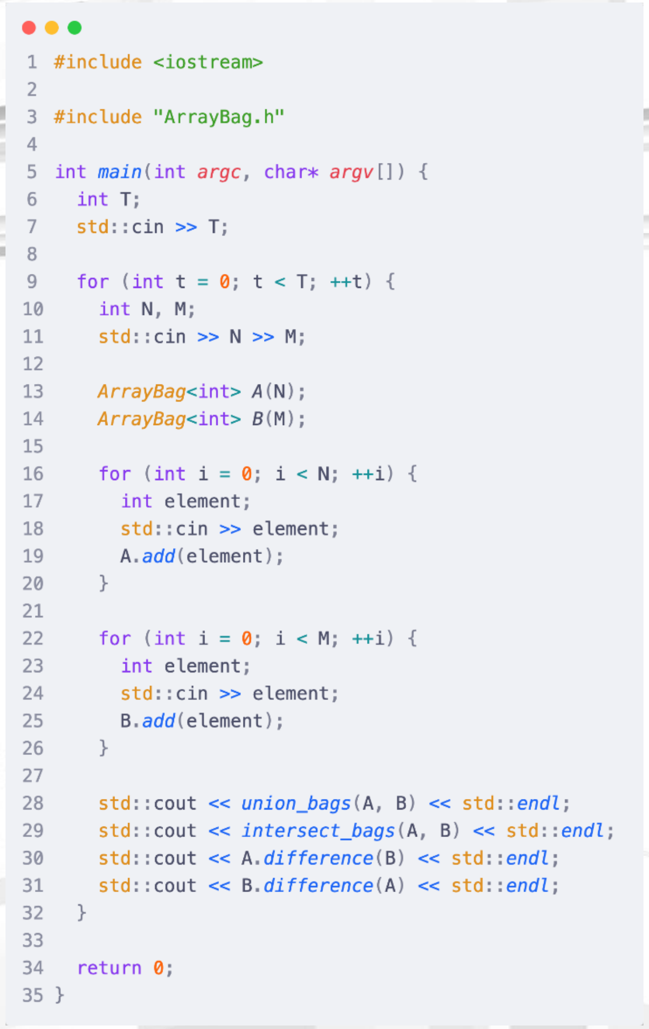


ArrayBag



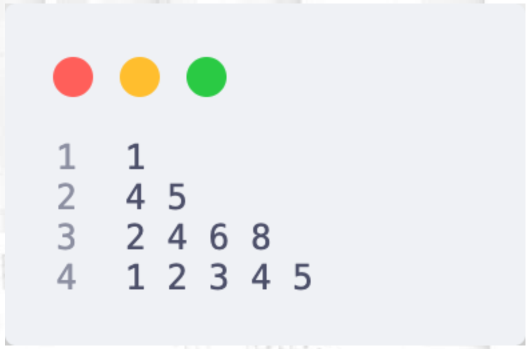
```cpp
#ifndef ARRAY_BAG_H
#define ARRAY_BAG_H
#include <iostream>
#include <vector>

template <typename T>
class ArrayBag {
 private:
  static const int DEFAULT_CAPACITY = 100;
  T items[DEFAULT_CAPACITY];
  int item_cnt;
  int max_items;
  int get_index_of(const T&) const;

 public:
  ArrayBag();
  ArrayBag(int);
  int get_current_size() const;
  bool is_empty() const;
  bool add(const T&);
  bool remove(const T&);
  void clear();
  bool contains(const T&) const;
  int get_frequency_of(const T&) const;
  std::vector<T> to_vector() const;
  template <typename U>
  friend std::ostream& operator<<(std::ostream&, const ArrayBag<U>&);
  template <typename U>
  friend ArrayBag<U> union_bags(const ArrayBag<U>&, const ArrayBag<U>&);
  template <typename U>
  friend ArrayBag<U> intersect_bags(const ArrayBag<U>&, const ArrayBag<U>&);
  ArrayBag<T> difference(const ArrayBag<T>&) const;
};

template <typename T>
ArrayBag<T> union_bags(const ArrayBag<T>&, const ArrayBag<T>&);
template <typename T>
ArrayBag<T> intersect_bags(const ArrayBag<T>&, const ArrayBag<T>&);

#include "ArrayBag.cpp"
#endif
```

main



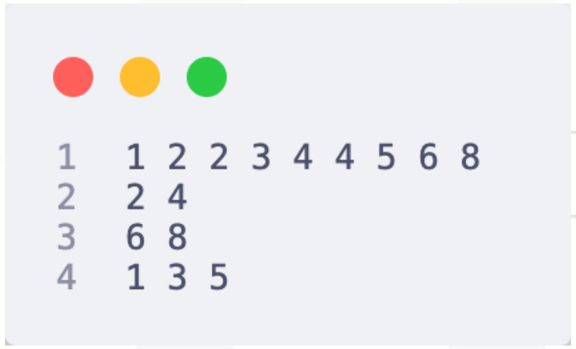
```cpp
#include <iostream>

#include "ArrayBag.h"

int main(int argc, char* argv[]) {
  int T;
  std::cin >> T;

  for (int t = 0; t < T; ++t) {
    int N, M;
    std::cin >> N >> M;

    ArrayBag<int> A(N);
    ArrayBag<int> B(M);

    for (int i = 0; i < N; ++i) {
      int element;
      std::cin >> element;
      A.add(element);
    }

    for (int i = 0; i < M; ++i) {
      int element;
      std::cin >> element;
      B.add(element);
    }

    std::cout << union_bags(A, B) << std::endl;
    std::cout << intersect_bags(A, B) << std::endl;
    std::cout << A.difference(B) << std::endl;
    std::cout << B.difference(A) << std::endl;
  }

  return 0;
}
```

실행 예시



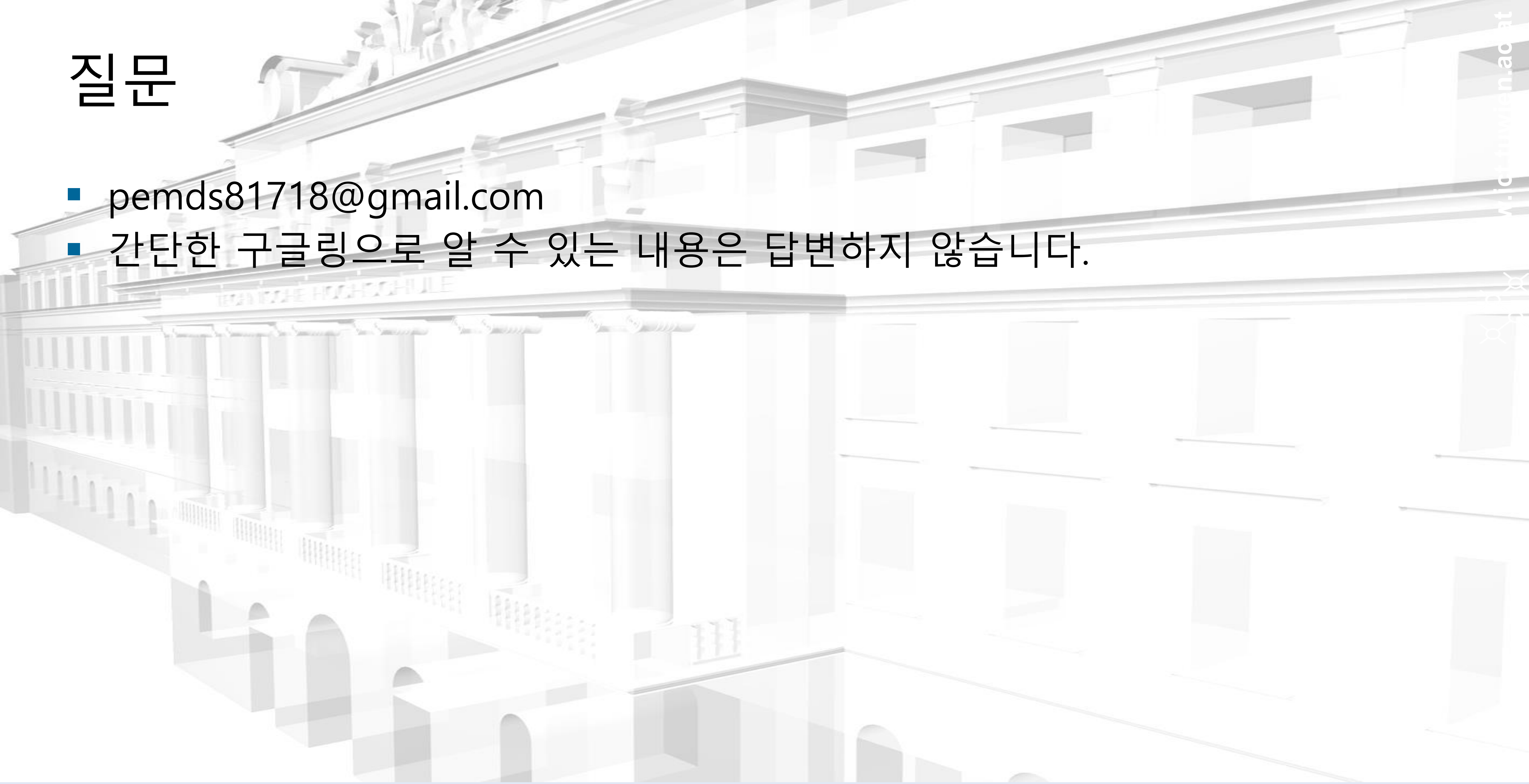
```
1
4 5
2 4 6 8
1 2 3 4 5
```

test.in



```
1 2 2 3 4 4 5 6 8
2 4
6 8
1 3 5
```

result.out



질문

- pemds81718@gmail.com
- 간단한 구글링으로 알 수 있는 내용은 답변하지 않습니다.