Robot Global Team

로봇 응용 SW 개발자 직무 면접 전 과제

㈜알지티

RGT

제출 방법

1. 소스코드 GIT링크, 터미널 출력 값, 실행 동영상 파일 첨부 필수입니다.
2. 다음 메일 주소로 송부합니다. rgt@rgt.kr

제출 기한

1. 배포일 기준 3일 입니다.

기타 사항

1. 모든 문제를 풀이 하지 않아도 괜찮습니다. 최대한 할 수 있는 만큼 해서 결과물을 제출하시면 됩니다.
2. 프로그램 제출시 github 제출 필수이고 github commit 규칙, 브렌치 관리 능력 또한 평가 대상입니다.
3. 프로그램 동작 환경 : window, visual studio code
4. 구현한 SW에 대한 설치 환경 정보에 대해서 명확하게 작성하여 제출 하십시오.
    1. 구현한 내용에 대해서 상세히 문서로 작성하여 제출해 주십시오(github 페이지 제출 필수, notion로 부가 설명 가능, notion 페이지는 경로는 대외적 공개는 불가능하고 RGT에만 제출).
    2. 동영상으로 해당 구현 결과를 상세하게 녹화 및 결과물을 캡쳐하여 제출 부탁드립니다. (ppt, notion 페이지 등등)

문제 : 스마트 포인터를 활용한 리소스 관리 : 터미널 출력값 캡쳐, 실행 동영상 제출 필수

1. 문제 설명
   1) 로그 파일들을 관리하는 LogFileManager 클래스를 구현
   2) 여러 로그 파일을 동시에 관리하며, 각 파일에 타임스탬프와 함께 로그를 기록할 수 있어야 함
2. 제출 필수 사항
   1. LogFileManager 클래스 정의 (public/private 접근 제한자 명시)
   2. 적절한 스마트 포인터 사용으로 파일 핸들 관리
   3. openLogFile(const std::string& filename) 메서드
   4. writeLog(const std::string& filename, const std::string& message) 메서드
   5. readLogs(const std::string& filename) → std::vector<std::string> 반환
   6. closeLogFile(const std::string& filename) 메서드
   7. 복사/이동 생성자 및 대입 연산자 적절히 처리
   8. 예외 안전성 보장 (파일 열기 실패, 쓰기 실패 등)

```cpp
// 제시한 변수명 변경하지 말 것
// 아래 내용을 구현하는 프로그램에 적절하게 설정할 것
LogFileManager manager;
manager.openLogFile("error.log");
manager.openLogFile("debug.log");
manager.openLogFile("info.log");

manager.writeLog("error.log", "Database connection failed");
manager.writeLog("debug.log", "User login attempt");
manager.writeLog("info.log", "Server started successfully");

std::vector<std::string> errorLogs = manager.readLogs("error.log");
```

입력 값

```
// error.log 파일 내용
[2025-09-04 14:30:15] Database connection failed

// debug.log 파일 내용
[2025-09-04 14:30:16] User login attempt

// info.log 파일 내용
[2025-09-04 14:30:17] Server started successfully

// readLogs 반환값
errorLogs[0] = "[2025-09-04 14:30:15] Database connection failed"
```

출력 값

문제 : 템플릿과 STL을 활용한 컨테이너 설계 : 터미널 출력값 캡쳐, 실행 동영상 제출 필수

1. 문제 설명
   1) 고정 크기 원형 버퍼 CircularBuffer<T>를 구현
   2) 센서 데이터 스트림을 효율적으로 저장하고 처리할 수 있어야 함
2. 제출 필수 사항
   1. 템플릿 클래스 CircularBuffer<T> 정의
   2. 생성자: CircularBuffer(size_t capacity)
   3. STL 호환 forward iterator 구현
   4. begin(), end(), size(), capacity(), empty() 메서드
   5. push_back(const T& item), pop_front(), front(), back() 메서드
   6. const와 non-const 버전 메서드 제공
   7. 범위 기반 for문 지원

```
// 제시한 변수명 변경하지 말 것
// 아래 내용을 구현하는 프로그램에 적절하게 설정할 것

// 테스트 시나리오
CircularBuffer<double> tempBuffer(5);

// 온도 센서 데이터 추가
tempBuffer.push_back(23.5);
tempBuffer.push_back(24.1);
tempBuffer.push_back(23.8);
tempBuffer.push_back(25.2);
tempBuffer.push_back(24.7);
tempBuffer.push_back(26.1); // 버퍼가 가득참 - 가장 오래된 값(23.5) 오버라이트

// STL 사용
double maxTemp = *std::max_element(tempBuffer.begin(), tempBuffer.end());
double avgTemp = std::accumulate(tempBuffer.begin(), tempBuffer.end(), 0.0) / tempBuffer.size();
```

입력 값

```
버퍼 내용 (인덱스 순서): [26.1, 24.1, 23.8, 25.2, 24.7]
begin()부터 순회 시: 24.1, 23.8, 25.2, 24.7, 26.1 (가장 오래된 것부터)

tempBuffer.size() = 5
tempBuffer.capacity() = 5
tempBuffer.empty() = false
maxTemp = 26.1
avgTemp = 24.78
tempBuffer.front() = 24.1  // 가장 오래된 데이터
tempBuffer.back() = 26.1   // 가장 최근 데이터
```

출력 값

문제 : 멀티스레딩과 함수형 프로그래밍을 활용한 병렬 처리 : 터미널 출력값 캡쳐, 실행 동영상 제출 필수

1. 문제 설명
   1) 대용량 이미지 데이터 처리를 위한 ParallelProcessor<T> 클래스를 구현
   2) 픽셀 데이터에 다양한 필터를 병렬로 적용
2. 제출 필수 사항
   1. 템플릿 클래스 CircularBuffer<T> 정의
   2. 생성자: CircularBuffer(size_t capacity)
   3. STL 호환 forward iterator 구현
   4. begin(), end(), size(), capacity(), empty() 메서드
   5. push_back(const T& item), pop_front(), front(), back() 메서드
   6. const와 non-const 버전 메서드 제공
   7. 범위 기반 for문 지원

```cpp
// 제시한 변수명 변경하지 말 것
// 아래 내용을 구현하는 프로그램에 적절하게 설정할 것

// 1000x1000 이미지의 픽셀 값들 ( 가상 설정 )
std::vector<int> pixelData(1000000);
std::iota(pixelData.begin(), pixelData.end(), 0);

ParallelProcessor<int> processor(4); // 4 쓰레드 설정 값 사용

// 밝기 증가 (각 픽셀값에 50 증가)
auto brightenedImage = processor.parallel_map(pixelData, [](int pixel) {
    std::this_thread::sleep_for(std::chrono::microseconds(1));
    return std::min(255, pixel + 50); // 이미지 픽셀값은 최대가 255임
});

auto pixelStrings = processor.parallel_map(pixelData, [](int pixel) -> std::string {
    return "pixel_" + std::to_string(pixel);
});

auto squaredPixels = processor.parallel_map(pixelData, [](int pixel) {
    return pixel * pixel;
});
```

입력 값

```
// brightenedImage 결과
brightenedImage[0] = 50     // 0 + 50
brightenedImage[1] = 51     // 1 + 50
brightenedImage[100] = 150  // 100 + 50
brightenedImage[999999] = 255 // min(255, 999999 + 50)

// pixelStrings 결과
pixelStrings[0] = "pixel_0"
pixelStrings[1] = "pixel_1"
pixelStrings[100] = "pixel_100"

// squaredPixels 결과
squaredPixels[0] = 0
squaredPixels[1] = 1
squaredPixels[10] = 100

// 성능 측정 결과 출력
Processing 1,000,000 elements with 4 threads
Sequential time: ~1000ms
Parallel time: ~250ms
Speedup: 4x
```

출력 값

Python 문제: Flask/FastAPI를 활용한 RESTful API 서버 구현 : 터미널 출력값 캡쳐, 실행 동영상 제출 필수

1. 문제 설명
   1) 온라인 도서관 관리 시스템의 백엔드 API를 구현
   2) 사용자 인증, 도서 관리, 대출/반납 기능을 제공하는 RESTful API 서버를 작성해야 함
2. 제출 필수 사항
   1. API 엔드포인트 구현: POST, GET, DELETE 부분 구현 내용
   2. 데이터베이스 모델
   3. 인증/인가 시스템
   4. 데이터 검증
   5. 환경설정 등등
3. 입력 값은 다음장에 설명

Python 문제: Flask/FastAPI를 활용한 RESTful API 서버 구현 : 터미널 출력값 캡쳐, 실행 동영상 제출 필수

```
import requests

base_url = "http://localhost:8000"

signup_data = {
    "username": "john_doe",
    "email": "john@example.com",
    "password": "securepass123",
    "full_name": "John Doe"
}
response = requests.post(f"{base_url}/auth/signup", json=signup_data)

login_data = {"username": "john_doe", "password": "securepass123"}
auth_response = requests.post(f"{base_url}/auth/login", json=login_data)
token = auth_response.json()["access_token"]
headers = {"Authorization": f"Bearer {token}"}

book_data = {
    "title": "Python Programming",
    "author": "John Smith",
    "isbn": "978-0123456789",
    "category": "Programming",
    "total_copies": 5
}
requests.post(f"{base_url}/books", json=book_data, headers=admin_headers)

search_response = requests.get(f"{base_url}/books?category=Programming&available=true")

borrow_data = {"book_id": 1, "user_id": 1}
requests.post(f"{base_url}/loans", json=borrow_data, headers=headers)

loans_response = requests.get(f"{base_url}/users/me/loans", headers=headers)
```

입력 파이선 파일

We provide good memories through our SIRBOT

RGT