

# System Description

```
// 출력 포맷 지정을 위한 라이브러리
#include <iomanip>

// EnumProcesses()를 사용하기 위한 헤더 파일, 라이브러리 링크
#define PSAPI_VERSION 1 // OS 문제 해결을 위한 옵션 define ...
#include <windows.h>
#include <psapi.h>
#pragma comment( lib, "Psapi.lib" ) // #pragma comment(lib, "라이브러리.lib") -> 명시적 라이브러리 링크
```

프로세스 개수를 가져오기 위해 psapi.h 라이브러리를 불러왔습니다.

이외에도 상단에 생략된 오류 방지 코드와 표준입출력 라이브러리를 불러왔습니다.

시간 출력 함수와 프로세스 개수 출력 함수를 분리하여 모듈화하였고,

main() 함수에서 호출하여 사용합니다.

## System Description

```
// 현재 시간을 출력하는 함수
void printTime() {
    SYSTEMTIME It;
    // GetSystem() -> UTC 기준 ... 따라서 GetLocalTime()을 사용해 시간을 가져옴
    GetLocalTime(&It); // get local time
    cout << It.wYear << "-";
    cout.width(2);
    cout.fill('0');
    cout << It.wMonth << "-";
    cout.width(2);
    cout << It.wDay << " ";
    cout.width(2);
    cout << It.wHour << ":";
    cout.width(2);
    cout << It.wMinute << ":";
    cout.width(2);
    cout << It.wSecond << " -> ";
}
```

GetSystem()은 UTC 기준의 시간을 출력하기 때문에 GetLocalTime()을 사용하였습니다.

SYSTEMTIME It 변수를 참조 변수로 받아 할당한 뒤 wYear, wMonth, ... 등으로 날짜와 시간을 각각 출력하였습니다.

## System Description

```
// 현재 프로세스 수를 반환하는 함수
int getProcessNum() {
    DWORD aProcesses[1024];
    DWORD cbNeeded;

    // processes가 없을 경우 return 1 ( 오류 )
    if (!EnumProcesses(aProcesses, sizeof(aProcesses), &cbNeeded)) {
        return 1;
    }

    return cbNeeded / sizeof(DWORD); // 프로세스의 개수 반환
}
```

EnumProcesses() 함수는 pid 리스트를 담을 배열과 그 배열의 사이즈를 전달 받아,  
그리고 참조 변수를 통해 aProcesses 배열의 byte수를 return합니다.

이를 DWORD의 사이즈로 나누어 개수를 구할 수 있습니다.

## System Description

```
// 현재 프로세스 수를 반환하는 함수
int getProcessNum() {
    DWORD aProcesses[1024];
    DWORD cbNeeded;

    // processes가 없을 경우 return 1 ( 오류 )
    if (!EnumProcesses(aProcesses, sizeof(aProcesses), &cbNeeded)) {
        return 1;
    }

    return cbNeeded / sizeof(DWORD); // 프로세스의 개수 반환
}
```

위와 같이 main() 함수에서 Sleep(1000)을 통해 1초 간격으로 함수를 호출하여 사용합니다.

# Test Description

```
C:\> C:\Users\user\source\repos\process_counter\x64\Debug\process_counter.exe
```

```
2024-04-08 17:08:07 -> 181
2024-04-08 17:08:08 -> 182
2024-04-08 17:08:10 -> 185
2024-04-08 17:08:11 -> 193
2024-04-08 17:08:12 -> 193
2024-04-08 17:08:13 -> 194
2024-04-08 17:08:14 -> 194
2024-04-08 17:08:15 -> 196
2024-04-08 17:08:16 -> 196
```

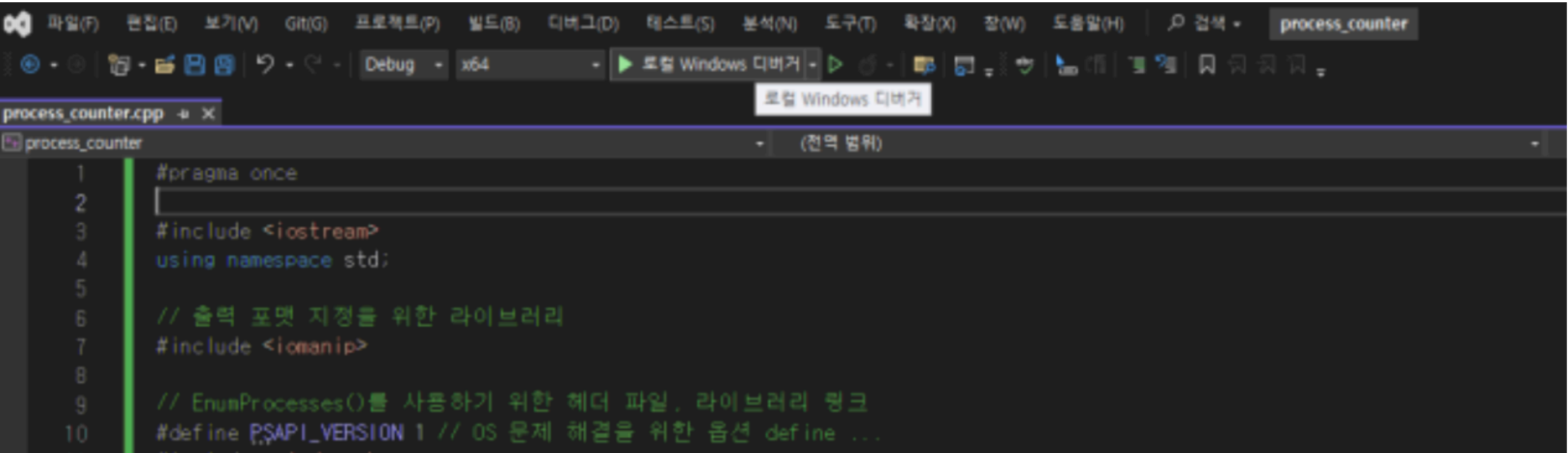
과제 예시와 다르게 가독성을 위해 :을 ->로 변경하였습니다.

1초 간격으로 프로세스 수가 정상적으로 출력되는 것을 확인할 수 있습니다.

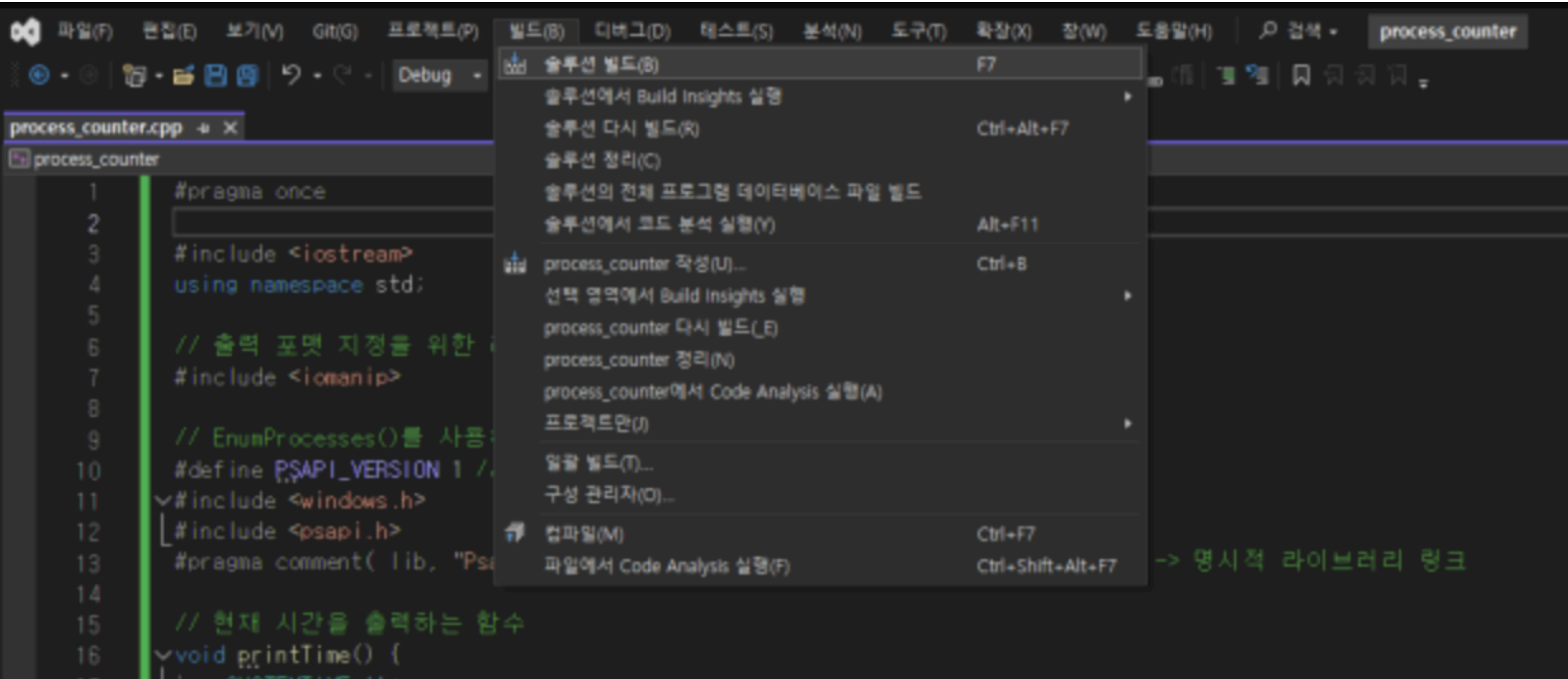


# User Documentation

- 1. .exe 파일 실행 ( 빌드된 상태 )
- 2. 로컬 디버거로 실행 : 프로젝트 생성 후 소스 파일에 추가 후 로컬 디버거로 실행



- 3. 빌드 후 .exe 파일 실행 : 마찬가지로 프로젝트 생성 후 빌드 - 솔루션 빌드 후 .exe 파일을 찾아 실행



| user > source > repos > process_counter > x64 > Debug |                    |                   |         |
|-------------------------------------------------------|--------------------|-------------------|---------|
| 이름                                                    | 수정한 날짜             | 유형                | 크기      |
| process_counter                                       | 2024-04-08 오후 5:08 | 응용 프로그램           | 73KB    |
| process_counter.pdb                                   | 2024-04-08 오후 5:08 | Program Debug ... | 1,348KB |

종 프로그