



[13주차 2강]

디버깅



학습 내용

- 디버깅의 기초



학습 목표

- 디버깅의 개념을 이해하고 기초적인 수준의 디버깅 요령을 익힌다.



1. 프로그램 오류의 종류와 디버깅



프로그램 오류의 종류와 디버깅

버그(bug)

- 프로그램에 존재하는 오류

디버깅(debugging)

- 오류를 고치는 행위

컴파일 오류

- 문법적 오류
- 컴파일러는 프로그램의 구문, 데이터, 의미 없는 문장 등을 검사
※ **컴파일 경고**: 오류는 아니지만, 오류의 가능성이 있는 부분

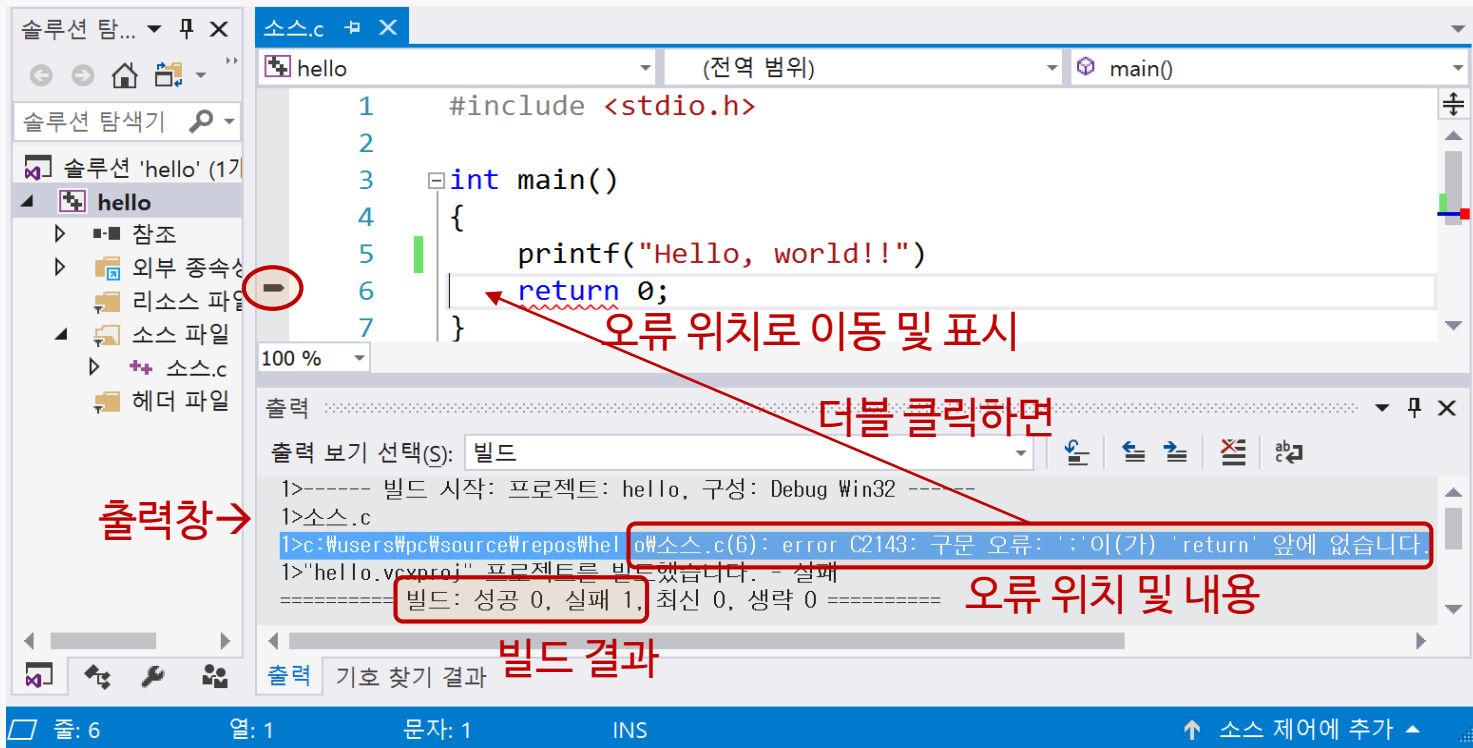
런타임 오류

- 실행시간 오류
- 프로그램의 결과가 의도와 다르거나 비정상적으로 종료되는 경우



컴파일 오류/경고 확인 및 수정

- 오류/경고의 위치와 종류를 컴파일러가 다 알려줌



출력창 →

오류 위치로 이동 및 표시

더블 클릭하면

오류 위치 및 내용

빌드 결과

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello, world!!")
6      return 0;
7  }
```

출력 보기 선택(S): 빌드

1>----- 빌드 시작: 프로젝트: hello, 구성: Debug Win32 -----

1>소스.c

1>c:\Users\wpc\source\repos\hello\소스.c(6): error C2143: 구문 오류: ':'이(가) 'return' 앞에 없습니다

1>"hello.vcxproj" 프로젝트를 빌드했습니다. - 실패

===== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 =====

출력 기호 찾기 결과

줄: 6 열: 1 문자: 1 INS

↑ 소스 제어에 추가



컴파일 오류/경고 확인 및 수정

- 컴파일 오류/경고를 고치는 과정은 매우 쉬움
 - ✓ 컴파일 오류/경고의 내용을 이해할 수 있어야 함



컴파일 오류 수정 Tip

- 오류로 표시된 라인과 실제 오류가 있는 라인이 다를 수 있음
 - ✓ 예) 세미콜론 누락, 중괄호 오류
- 가장 먼저 표시되는 오류 내용부터 확인하고 수정
 - ✓ 실제 오류는 하나이지만, 이로 인해 여러 군데 오류가 발생하는 것처럼 보일 수 있음
 - ✓ 맨 처음 표시되는 오류 내용을 해결하면, 나머지는 저절로 해결되는 경우 많음

2. 프로그램 오류 비교




컴파일 오류/경고와 빌드(실행파일 생성)

- 컴파일 오류가 있으면 실행 파일이 생성되지 않음
- 컴파일 경고는 있어도 실행 파일 잘 생성 됨
 - ✓ 경고는 문법적으로 오류가 없어 기계어로 번역하는데 문제는 없으나, 오류 가능성이 있거나 권장되지 않은 방법으로 작성된 부분



VS 사용 시 주의 사항

- VS에서 프로그램을 수행(단축키: [Ctrl+F5])시키면
 - ✓ (소스가 변경된 경우) 프로그램 빌드 → 프로그램 수행
 - ✓ 만약 빌드(실행파일 생성)에 실패하면, 기존 실행 파일 수행
- 귀찮다고 무작정 [Ctrl+F5] 누르지 말고, 빌드를 먼저 수행하여 오류나 경고가 없는 지 확인하자!!

- 
- ### 런타임 오류 확인 및 수정
- 프로그램 실행 시 발생하는 오류
 - 일반적으로 오류가 어디서 발생했는지 알기가 어려움
 - 대부분의 시간을 이 오류를 찾고 수정하는데 소비
 - 이러한 종류의 오류를 찾아 고치는 작업 → 디버깅(debugging)

어떻게 하면 런타임 오류를 쉽게 찾아 고칠 수 있을까?
(디버깅의 기본)

3. 디버깅 방법



방법 1

- 프로그램 소스를 쭉 보면서 내가 어디서 틀렸나 찾아본다.
- 한계점
 - ✓ 코드가 짧은 경우에 유효
 - ✓ 코드가 긴 경우 못 찾을 가능성이 훨~~씬 큼



방법 2

- 실제로 프로그램을 수행시켜보면서
- 프로그램이 내가 예상하고 원하는 과정대로 동작하는 지를 체크



프로그램 테스트 방법

- 여러분은 어떻게 프로그램이 올바르게 동작하는 지를 테스트하는가?
 - ✓ 실제로 실행 시켜 보고, 결과를 확인한다.
- 예를 들어, 어떤 복잡한 계산을 하는 과학기술용 프로그램이라면?
 - ✓ 계산된 결과가 맞는 지 체크
 - ✓ 계산된 결과는 어떻게 알 수 있나?
 - 가장 간단한 방법은 화면에 결과 값을 출력

3. 디버깅 방법



프로그램의 **중간 과정** 테스트는?

- 즉, 프로그램이 내가 원하는 과정대로 동작하고 있는가?
 - ✓ **제어 흐름**과 **변수 값** 변화
- 앞의 예(복잡한 계산을 하는 과학 기술용 프로그램이라면?)
 - ✓ 중간에 계산되는 값들이 맞는 지 체크.
 - ✓ 중간 계산 결과를 어떻게 알 수 있나?
 - 가장 간단한 방법은 화면에 결과 값을 출력

3. 디버깅 방법



디버깅의 기본!!!

- 코드 중간에 있는 오류를 찾기 위해서
- 프로그램이 내가 의도하는 (중간) 과정대로 동작하는 지를 체크
- 무엇을 체크?
 - ✓ 제어 흐름
 - ✓ 변수에 저장된 값 및 (주소)를 확인
- 어떻게 체크?
 - ✓ 가장 간단한 방법은 화면에 결과 값을 출력해서 확인
 - ✓ 코드에 값 체크용 조건문을 넣어서 확인

※ 개발 툴에서 제공되는 디버깅 기능 활용 (추후에)



[예제 1] 1~10까지의 합 계산

```
int main( ) {  
    int i, sum;  
  
    sum = 0;  
    for( i=1; i < 10 ; ++i )  
    {  
        sum = sum + i++;  
    }  
  
    printf("sum = %d\n", sum);  
}
```

결과는?



[예제 1]

1~10까지의 합 계산 - 중간 결과 체크



```
int main( ) {  
    int i, sum;  
  
    sum = 0;  
    for( i=1; i < 10 ; ++i )  
    {  
        sum = sum + i++;  
        printf("i = %d, sum = %d\n", i, sum);  
    }  
  
    printf("sum = %d\n", sum);  
}
```

결과는?



디버깅 요령

- 어느 부분에서 체크할 것인가? → 프로그램의 논리 구조와 관계
- 프로그램을 몇 가지 단계로 나누어 단계별로 체크
 - ✓ 단계별로 체크하면서 작성하는 것도 좋은 방법
- 오류가 있는 범위를 좁혀 나가기
- 주요 체크 부분
 - ✓ 입력이 제대로 저장됐는지 체크
 - ✓ 반복문의 경우, 반복문의 시작과 끝
 - ✓ 함수의 시작과 끝



반복문 체크 요령

- 1) 반복문 전 후에 체크
- 2) 반복문 후에 오류가 발생했다면, 반복문 내부 체크
✓ 반복문 시작 시점 또는 끝 지점에서 체크

[check (1)] → 여기까지는 오류 없이 잘 실행됨

```
while( ... ) {  
    [check (2)] → 반복문 내부 체크 지점  
    ...  
    [check (2)] → 반복문 내부 체크 지점  
}
```

[check (1)] → 여기서 결과에 오류가 있음



함수 체크 요령

- 1) 함수 호출 전 후에 체크
- 2) 함수 호출 후에 오류가 발생했다면, 함수 내부 체크
 - ✓ 함수 시작 시점과 끝 지점에서 체크

```
void func2(..){  
    [check (2)] → 함수 내부 체크 지점  
    ...  
    [check (2)] → 함수 내부 체크 지점  
}  
void func1(..){  
    ...  
    [check (1)] → 여기까지는 오류 없이 잘 실행됨  
    func2(...);  
    [check (1)] → 여기서 결과에 오류가 있음  
    ...  
}
```

4. 버그와 런타임 오류

프로그래밍 초심자가 많이 겪는 상황, 또는 의문

- VS에서 잘 동작하는데, OJ에서 제대로 동작하지 않아요
- OJ에 뭔가 문제가 있는 거 아니가요?
- 결론부터 말하면,
 - ✓ 코드에 버그가 있는 경우가 대부분
- 그럼, 왜 VS에서는 잘 동작하나요?
- 버그가 있으면, 반드시 런타임 오류가 발생하나요?

4. 버그와 런타임 오류

 버그가 있는데도 정상 동작하는 상황 예시

 잘 동작하는 데이터로만 테스트 하는 경우

- (예시) 90점 이상을 A학점으로 판별하는 코드

```
if( score > 90 ) printf("A");  
else if( score >= 80) printf("B");  
...
```

- ✓ 다음 score 값으로만 테스트를 했다면?
 - ✓ 100, 95, 88
- ✓ 버그가 있음에도, 위 예에 대해서는 정상 동작

4. 버그와 런타임 오류

 버그가 있는데도 정상 동작하는 상황 예시

 우연히 잘 동작하는 경우도 있음

- (예시) 크기가 5인 배열 A의 모든 원소의 합을 구하는 코드

```
for( i=0 ; i <= 5 ; ++i )  
    sum += A[i];
```

- ✓ sum의 결과는? $A[0] + A[1] + A[2] + A[3] + A[4] + A[5]$
- ✓ 우연히 A[5] 자리의 변수 값이 0이라면, 정상 동작

4. 버그와 런타임 오류



버그가 있는데도 정상 동작하는 상황 예시



환경에 따라 잘 동작하는 경우도 있음

- 환경: 컴파일러, 하드웨어, OS 등
- 환경에 따라, 메모리 관리 등이 다를 수 있음
- 표준에 정해지지 않은 사항은 컴파일러에 따라 동작이 다를 수 있음
- VS는 버그가 있어도 동작하도록 표준에 없는 사항을 적용한 경우가 많음
- 버그로 인해 gcc(OJ의 컴파일러)에서는 오류가 발생하는데,
VS에서는 정상 동작하는 경우가 있음


4. 버그와 런타임 오류



버그와 런타임 오류 정리

- 버그가 없는 코드
 - ✓ 어떤 실행 환경, 어떤 테스트 데이터 든지 모두 정상 동작함
- 버그가 있는 코드
 - ✓ 상황에 따라, 정상 동작하는 경우 처럼 보일 수 있음
- 따라서, 정상 동작하지 않는 경우가 있다면,
코드에 버그가 있다는 의미

4. 버그와 런타임 오류

 VS에서는 정상 동작하는데, OJ에서는 오류가 난다면 어떻게 디버깅을 해야 할까?

- 다양한 데이터를 만들어서 테스트
 - ✓ 경계 조건에 해당하는 데이터
 - ✓ 모든 경우를 잘 반영할 수 있는 데이터 셋
- OJ와 동일한 환경(우분투, gcc)에서 테스트?
 - ✓ 컴파일러 만이라도 동일
 - ✓ gcc를 직접 설치하기 어려운 경우, 온라인 컴파일러 이용

학습 정리

- 프로그램에 존재하는 오류를 **버그**라고 하고 이를 찾아서 올바르게 고치는 것을 디버깅이라 함
- 디버깅을 위해서는 프로그램이 내가 의도하는 **(중간) 과정대로 동작**하는 지를 체크해야 함
- 디버깅시 확인해야 할 것은 **제어 흐름과 변수의 값**임
- 코드에 버그가 있어도, 상황에 따라 정상 동작하는 경우도 있음