C 프로그래밍 및 실습

부록. VS와 디버깅

세종대학교

목차

Visual Studio 이해하기(1) (2010 기준)

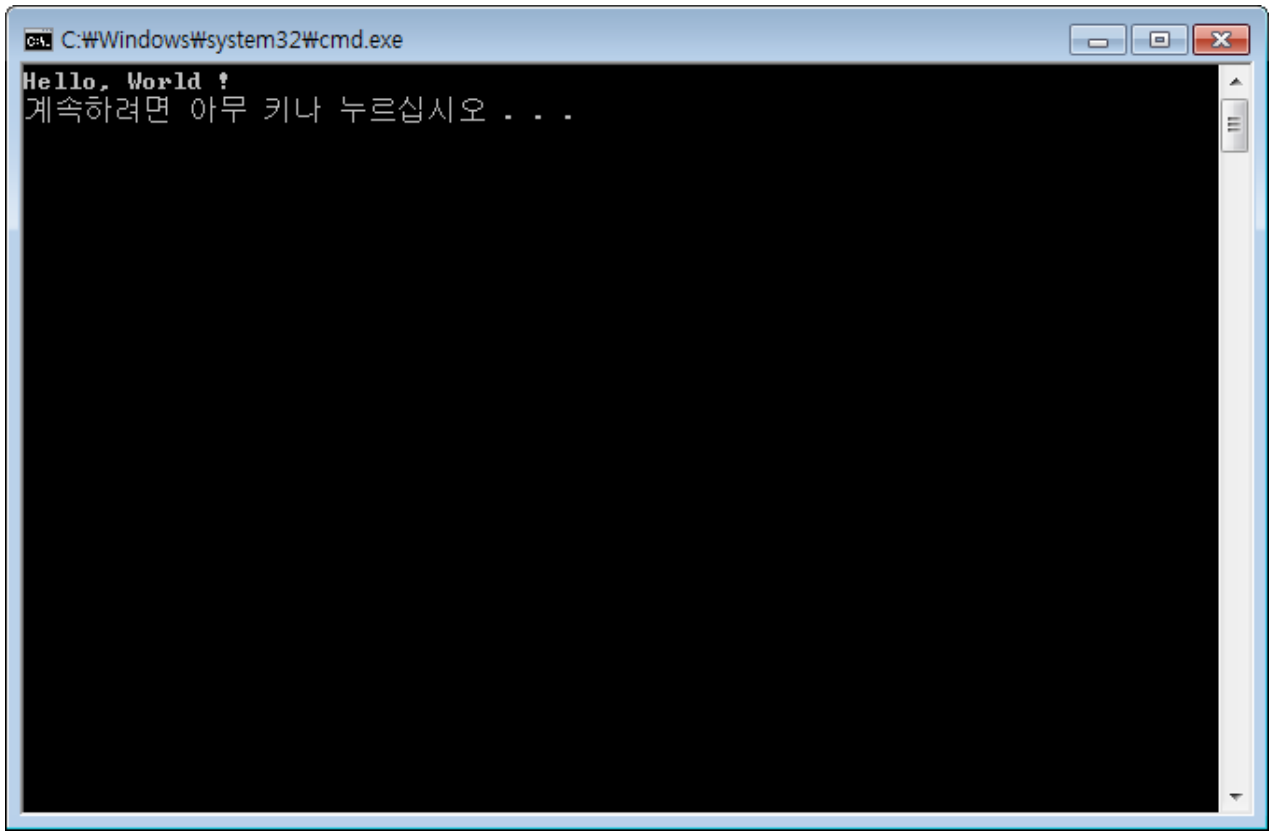
- **프로젝트**
  - 하나의 실행파일을 제작하는데 필요한 관련 파일의 집합
  - 보통 여러 개의 헤더, 소스, 리소스 파일들로 구성
- **솔루션**
  - 여러 개의 프로젝트들의 집합 → 여러 개의 실행파일 (제품)
- **비유**
  - MS-Office : 솔루션
  - 워드, 파워포인트 등: 프로젝트들
- **프로젝트, 솔루션이라는 개념은 C언어와 무관**

Visual Studio 이해하기(2)

- **다음 순서대로 hello world! 프로그램 작성**
  1. Visual Studio 실행 (2010 버전 기준)
  2. **C:₩**에 "hello-s" 솔루션과 "hello-p" 프로젝트 생성한 후, 탐색기를 이용하여 C:₩hello-s 폴더의 파일들 확인
  3. hello.c 소스 파일 추가 및 코딩 ➔ 폴더의 파일들 확인
  4. 솔루션 빌드 ➔ 폴더의 파일들 확인
     - ✓ c:₩hello-s₩Debug 폴더 안에 실행파일 hello.exe
     - ✓ c:₩hello-s₩hello-p₩Debug 폴더의 파일들
  5. 솔루션 정리 ➔ 폴더의 파일들 확인

Visual Studio 이해하기(3)

- **우리가 작성한 프로그램은?**
  - 콘솔 화면(검은색 창)에 **Hello World !**라고 출력하는 프로그램



Visual Studio 이해하기(4)

- **작성된 프로그램과 일반 프로그램(메모장)과의 비교**

Hello World!	메모장
콘솔 프로그램	윈도우 프로그램
VS 상에서 실행	VS가 없어도 실행 가능

- 우리가 작성한 Hello World! 프로그램은 **VS 없이는 실행**할 수 없는가? **아니오!! VS 없이도 실행 가능**

Visual Studio 이해하기(5)

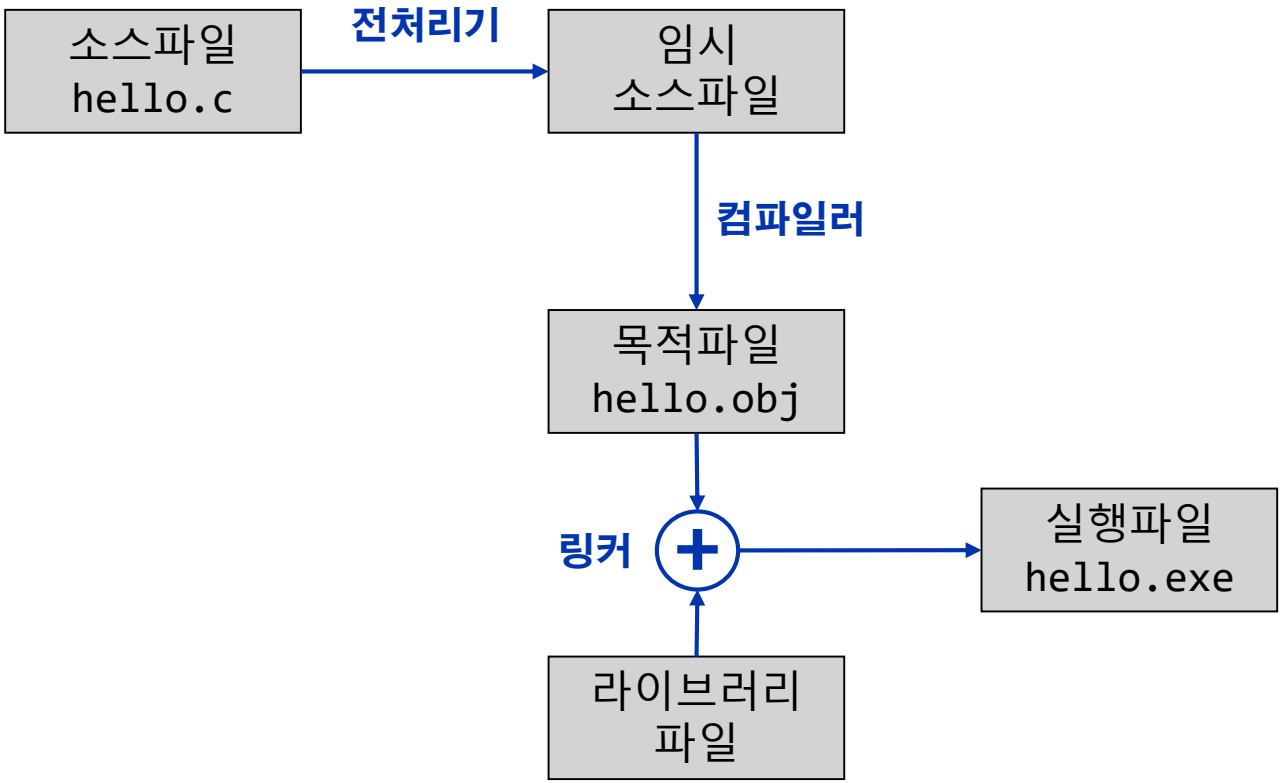
- **Visual studio의 역할**
  - 소스파일(텍스트 파일)로 부터
    - ✓ 메모장으로 열어서 확인해 보자.
      - ✓ VS에서 열어본 hello.c와의 차이점은?
  - 실행파일 (.exe)을 만들어 주는 tool일 뿐
    - ✓ VS를 종료하고 탐색기에서 .exe 파일 클릭하면?
  - 부가적으로, 프로그램 작성을 도와주는 기능 포함
    - ✓ 디버거
    - ✓ 프로젝트, 솔루션

CMD 모드

- **CMD (명령프롬프트)**
  - 일종의 text 기반 사용자 interface (명령창에서 cmd 실행)
  - 폴더 경로 및 파일 확인해보기
  - 기본 명령어
    - ✓ cd : 폴더 이동을 위한 명령어 (cd 경로이름, cd ₩, cd ..)
    - ✓ dir : 폴더의 파일 목록을 보여주는 명령어 (dir /w)
    - ✓ 탐색기와 비슷한 역할
- **CMD에서 hello world! 프로그램 학인**
  - 생성된 파일 확인
  - 실행파일(.exe) 실행해 보자
  - notepad.exe, calc.exe 도 실행해보자.

실행파일 만드는 과정

- 소스파일에서 실행파일까지



hello world!

- **hello world! 빌드 과정 세분화**

1. 솔루션 정리 후 "컴파일" 메뉴 실행 ➔ 폴더의 파일들 확인
   - ✓ 특히, Debug 폴더들의 파일 변화에 주의

2. 솔루션 빌드 ➔ 폴더의 파일들 확인
   - ✓ Debug 폴더 안에 파일들의 생성 시간을 확인해보자.

- 빌드 : 컴파일 + 링크

컴파일 vs. 빌드

- **컴파일**
  - 해당 소스파일을 기계코드(이진파일)로 변환하여
  - 목적 파일(.obj)을 생성
    - ✓ 목적 파일은 그 자체만으로는 실행되지 않음

- **링크**
  - 목적 파일, 라이브러리 파일들을 결합하고, 실행파일에 필요한 코드를 추가하여
  - 실행 파일(.exe)을 생성

- **목적 파일과 실행 파일의 크기를 비교해보자.**

기능 확인 (1)

- **main 함수가 존재하지 않는 소스**
  - 컴파일 해보자.
    - ✓ 에러 발생?
  - 빌드 해보자.
    - ✓ 에러 발생?
  - 결과를 보고 왜 그럴까 생각해보자.

기능 확인 (2)

- **어떤 함수가 선언만 되고, 정의되지 않은 소스**
  - 컴파일 해보자.
    - ✓ 에러 발생?
  - 빌드 해보자.
    - ✓ 에러 발생?
  - 정의되지 않은 함수를 사용한 소스를 컴파일 하면?

목차

프로그램 오류의 종류

- **컴파일 오류 (컴파일 시간 오류, 구문 오류, 문법 오류)**
  - 컴파일 과정에서 생긴 오류
  - 컴파일러는 프로그램의 구문, 데이터, 의미 없는 문장 등을 검사
  - 이런 경우 편집기로 돌아가서 오류를 수정한 후 다시 컴파일 해야 함

- **런타임 오류 (실행시간 오류)**
  - 프로그램실행 중에 예상치 못한 이유로 비정상적으로 종료
  - 예: 어떤 값을 0으로 나누려고 하면 프로그램은 실행을 멈추고 종료
  - 좋은 프로그램은 실행시간 오류가 가능한 한 발생하지 않아야 함

- **논리 오류**
  - 성공적인 컴파일 후 프로그램을 실행하면서 잘못된 결과를 내는 경우

프로그램 오류 비교

- **컴파일 오류**
  - 오류의 위치와 종류를 찾고 수정하는 과정이 매우 쉬움
  - 컴파일러가 다 알려줌

- **런타임 오류 & 논리 오류**
  - 일반적으로 오류가 어디서 발생했는지 알기가 어려움
  - 프로그램 작성하는 대부분의 시간을 이 오류를 찾고 수정하는데 소비
  - 이러한 종류의 오류를 찾아 고치는 작업 ➔ 디버깅(debugging)

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

디버깅 방법

- **방법 1**
  - 프로그램 소스를 쭉 보면서 내가 어디서 틀렸나 찾아본다.
  - 한계점
    - ✓ 소스가 짧은 경우에나 가능
    - ✓ 못 찾을 가능성이 훨~~씬 큼

- **방법 2**
  - 실제로 프로그램을 수행시켜보면서
  - 프로그램이 내가 예상하고 원하는 **과정대로** 동작하는 지를 체크

프로그램 테스트

- **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**

프로그램의 중간 과정 테스트

- **프로그램의 중간 과정 테스트는?**
  - 즉, 프로그램이 내가 원하는 **과정대로** 동작하고 있는가?

  - 앞의 예(복잡한 계산을 하는 과학기술용 프로그램이라면?)
    - ✓ **중간에 계산되는 값**들이 맞는 지 체크.
    - ✓ 중간 계산 결과는 어떻게 알 수 있나?
    - → 가장 간단한 방법은 **화면에 결과 값을 출력**

디버깅의 기본

- **디버깅의 기본!!!**
  - 중간의 오류를 찾기 위해서
  - 프로그램이 내가 예상하는 **(중간) 과정대로 동작**하는 지를 체크

  - 이를 체크하는 방법은?
    - ✓ 프로그램 실행 도중에 **변수에 저장된 값 및 주소**를 확인
    - ✓ 변수 값 및 주소 값은 어떻게 확인할 수 있나?
    - → 가장 간단한 방법은 화면에 결과 값을 출력

디버깅 연습

- **[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?

디버깅 연습

- [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 연습

- **[예제 2] 이름 출력**

```
void InputName(char *pName)
{
   pName = "Cabin";
}

int main( )
{  char *Name = NULL;

   InputName(Name);
   printf("Name : %s\n", Name);
}

결과는?
```

디버깅 연습

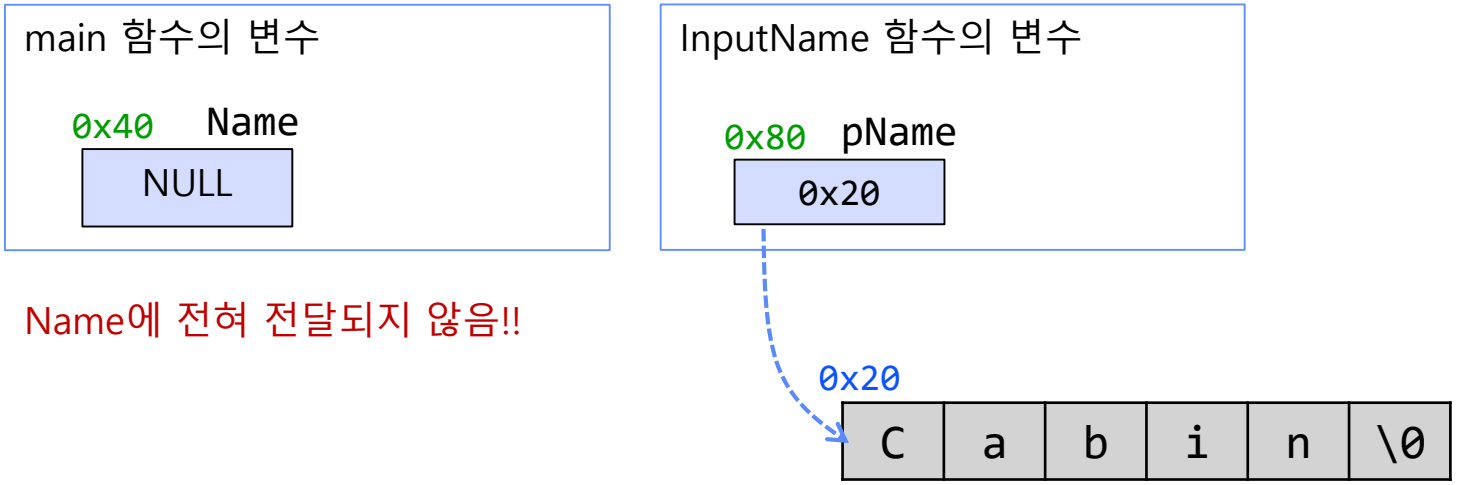
- [예제 2] 이름 출력 – 중간 결과 체크

```
void InputName(char *pName)
{  printf("InputName 1: pName = %p, %s\n", pName, pName);
   pName = "Cabin";
   printf("InputName 2: pName = %p, %s\n", pName, pName);
}

int main( )
{  char *Name = NULL;

   InputName(Name);
   printf("main: Name = %p\n", Name);
   printf("Name : %s\n", Name);
}

결과는?
```

Name에 저장된 값을 주소로 출력

예제2

- **결과는?**
  - InputName 함수의 pName의 값이 main 함수로 전달이 안됨?
    - ✓ 어라? 포인터를 쓰면 주소가 넘어가니까 전달이 되야 될 거 같은데…
    - ✓ 왜 안되지?
    - ✓ 메모리 그림을 통해 확인해보자!! (InputName 함수 종료 직전)
    - ✓ (생각해보기) main에 전달하려면 어떻게 수정해야 해야 할까?



Visual Studio의 디버거

- **값을 출력해서 중간 과정을 확인하는 것은 일반적으로 불편함**
  - ✓ 하지만, 값 출력하는 것이 편할 때도 있다...

➔Visual Studio에서는 보다 편리한 디버깅 기능 지원

- 변수의 값을 쉽게 확인
- 제어 흐름을 쉽게 따라갈 수 있음
- 기타 부가적인 기능을 제공..

- VS의 "디버그" 메뉴

VS 디버거의 기초

- 디버그/릴리즈 모드 (툴바에서 변경 가능)
  - 디버그 모드
    - ✓ 실행파일 만들 때 디버깅을 위한 정보도 같이 생성
    - ✓ 크기가 커지고, 느림
  - 릴리즈 모드
    - ✓ 디버깅 정보 없이 순수 실행 코드만 생성
    - ✓ 크기가 작고 빠름
  - VS를 이용한 프로그램 개발 순서
    - ✓ 디버그 모드로 프로그램을 개발하다,
    - ✓ 최종 제품은 릴리즈 모드로 프로그램 생성

  - 주의!! 디버그 모드와 릴리즈 모드는 메모리 관리가 다르기 때문에, 디버그 모드에서 잘 수행돼도, 릴리즈 모드에서 에러가 발생할 수 있다.

VS 디버거 사용법의 기초 (1)

- **디버깅 흐름 제어**
  - "디버그"메뉴 → 디버깅 시작(F5)
    - ✓ 무슨 일이 발생하는가?
  - 중단점(breakpoint) 설정/해제 (F9)
    - ✓ 프로그램 실행을 잠시 멈추고 싶은 지점(중단점) 설정
    - ✓ 소스에서 해당위치에 마우스 우클릭→ 중단점으로 지정
    - ✓ 또는 커서를 원하는 위치에 두고, "디버그"메뉴 → 중단점 설정 해제
  - 앞의 예에서 실습해봅시다.

VS 디버거 사용법의 기초 (2)

- 한 단계씩 코드 실행(F11)
  - ✓ 코드를 한 줄씩 실행하고, 함수 안으로 들어감.
  - ✓ 예제1에서 실습

- 프로시저 단위 실행(F10)
  - ✓ 코드를 단계적으로 실행하되, 함수를 하나의 단계로 처리
  - ✓ 예제2에서 실습

- 프로시저 나가기(Shift + F11)
  - ✓ 현재 함수의 끝까지 실행하고 함수 밖으로 나감

- 커서 위치까지 실행(Ctrl+ F10)
  - ✓ 커서의 위치까지 실행

VS 디버거 사용법의 기초 (3)

- **변수 값 확인**
  - 원하는 변수에 마우스 커서를 댄다.
    - ✓ 복잡한 형태의 변수이면 블록으로 선택 후 마우스

  - 변수의 값을 보여주는 아래쪽 창 확인
    - ✓ 자동
    - ✓ 지역