프로그래밍 기초

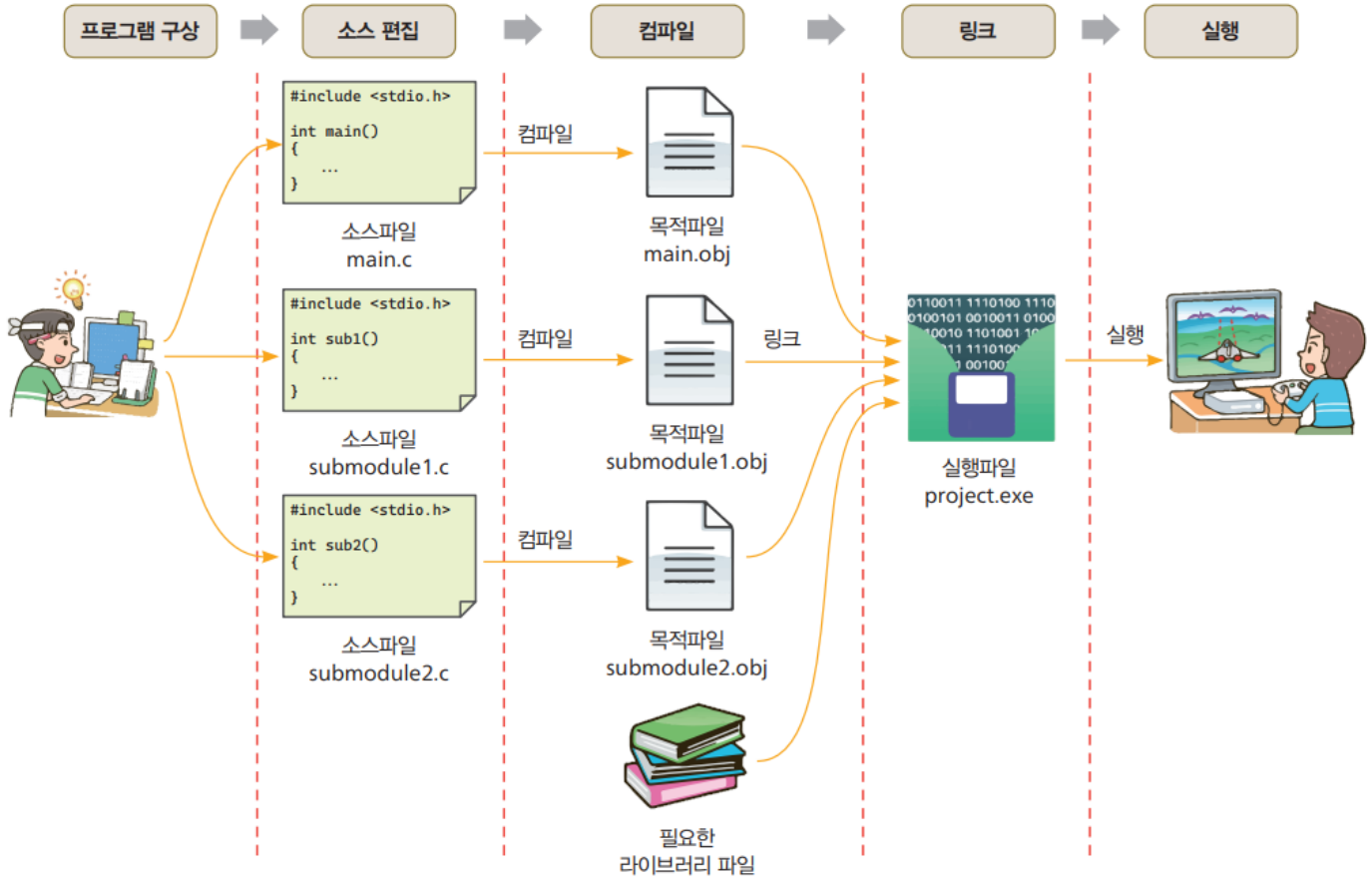
2장. C 프로그래밍 첫 걸음

학습 목표

- **프로그램 구현 과정과 통합개발환경을 설명할 수 있다.**
  - 프로그램 구상, 소스 편집, 컴파일, 링크, 실행 과정
  - 통합개발환경의 필요 소프트웨어
- **VS Code로 C 프로그램을 개발할 수 있다**
  - 소스 생성
  - 컴파일 및 실행
- **함수 printf()와 puts()로 구성되는 C 프로그램을 구현할 수 있다.**
- **오류의 종류와 원인 파악**
  - 오류 발생에 따른 디버깅 과정

프로그램 구현 과정

- **SW 개발 5단계**
  - 요구사항 분석, 설계, 구현, 검증, 유지 보수
- **프로그램 구현 절차**



소스 편집

- **소스 파일(souce file) 또는 소스 코드(source code)**
  - 프로그래밍 언어로 명령어가 저장된 파일
  - 텍스트 파일

- **소스 파일 확장자**
  - C언어: .c
  - 자바: .java
  - C++: .cpp

소스파일은 텍스트파일 형식이므로 모든 편집기로 읽거나 작성할 수 있으나 아래한글이나 워드와 같은 문서편집기에서는 반드시 텍스트 형태로 저장해야 한다.

```
# include <stdio.h>

int main()
{
   printf("hello, world\n");

   return 0;
}
```

C 소스파일: *.c

소스파일은 모든 편집기로 읽을 수 있으나 아래한글이나 워드로 작성된 일반 문서는 내부 형식이 다르므로 작성한 편집기로만 읽을 수 있다.

아래한글로 작성된 문서는 .hwp라는 확장자이듯, 소스파일은 프로그래밍 언어에 따라 고유한 확장자를 갖는데, C 언어는 .c이며 자바는 .java 그리고 C++는 .cpp이다.

아래한글 파일: *.hwp

4

컴파일

- **컴파일러(compiler)**
  - 소스 파일에서 기계어로 작성된 오브젝트 파일(object file)을 만들어내는 프로그램
    - 소스 파일이 main.c, submodule1.c, submodule2.c
    - 3개의 오브젝트 파일이 생성
      - main.o, submodule1.o, submodule2.o

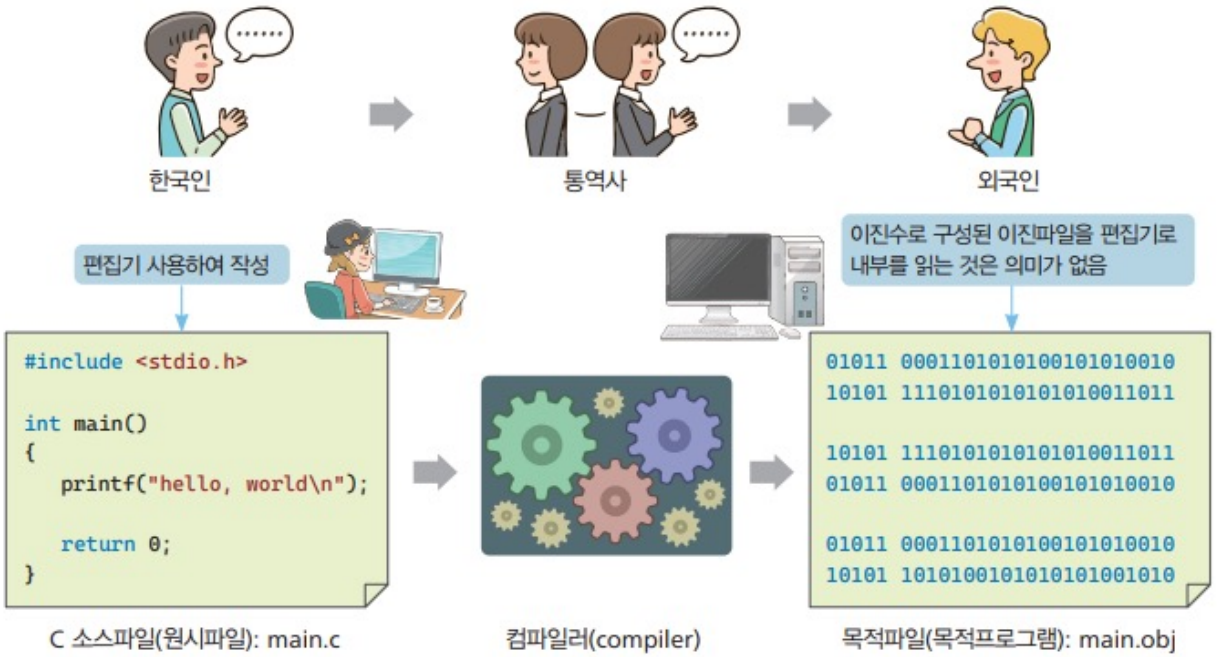


그림 2-3 컴파일의 이해

링크와 실행

- **링크(link) 또는 링킹(linking)**
  - 여러 개의 오브젝트 파일과 라이브러리를 하나의 실행 파일로 결합(연결)하는 과정

- **링커(linker)**
  - 하나 이상의 오브젝트 파일들을 하나의 실행 파일(execute file)로 만들어 주는 프로그램
    - 참조하는 여러 라이브러리를 포함시킴
    - 링크 결과인 실행 파일의 확장자는 .exe 또는 .dll, .com 등

- **라이브러리(library)**
  - 자주 사용하는 프로그램을 미리 만들어 놓고 사용하는 모듈
  - 개발자마다 새로 작성할 필요 없이 개발환경에서 미리 만들어 컴파일해 저장해 놓은 모듈

- **VSCode Run C/C++ File 메뉴**
  - 컴파일과 링크 과정을 하나로 합친 메뉴
  - 컴파일과 링크가 성공하면 실행 파일(.exe)이 생성되고, 실행됨

링크와 실행

- 링커 동작

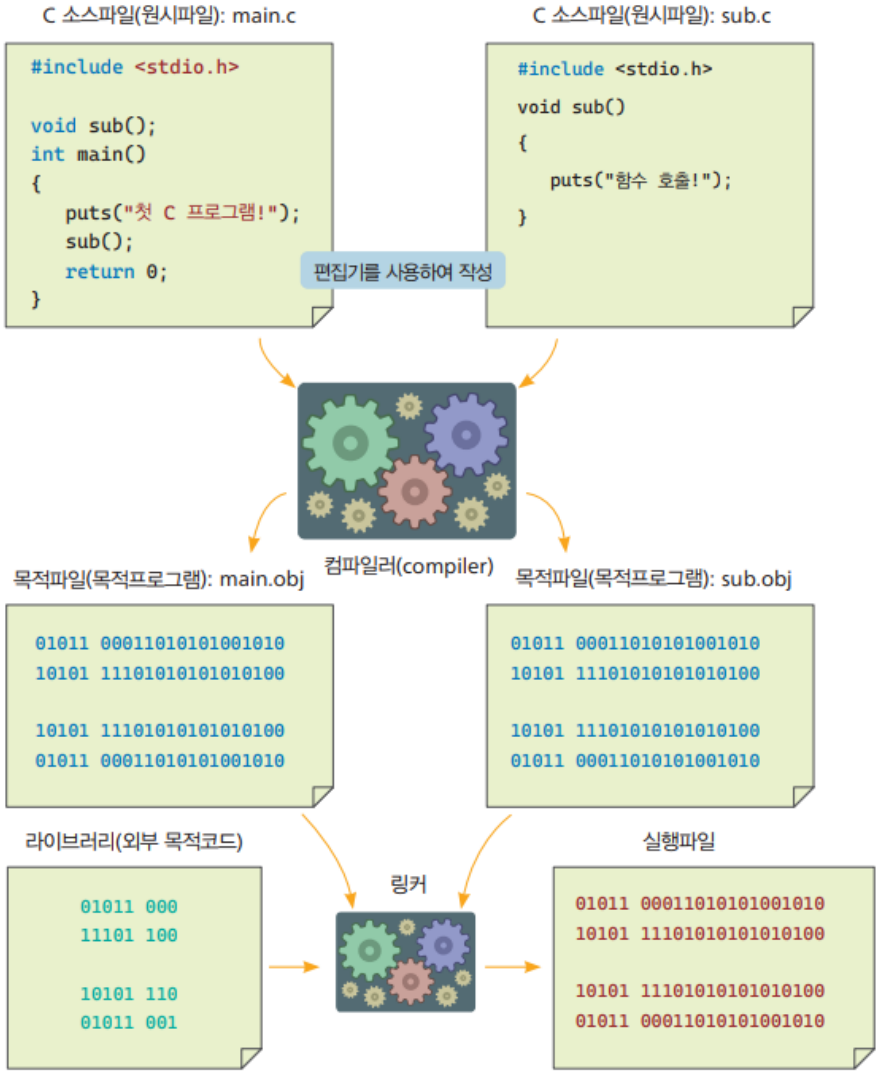


그림 2-4 링커의 이해

오류(Error)

- **오류 또는 에러(error)**
  - 프로그램 개발 과정에서 나타나는 모든 문제
  - 발생 시점에 따른 구분
    - **컴파일(시간) 오류**
      - 개발환경에서 오류 내용과 오류 발생 위치를 어느 정도 알려줌
      - 오류를 찾아 수정하기가 비교적 용이
    - **링크(시간) 오류**
      - 컴파일 오류보다 상대적으로 희소
      - main() 함수 이름이나 라이브러리 함수 이름을 잘못 기술하여 발생
    - **실행(시간) 오류**
      - 실행 도중 오류가 발생해서 실행이 중지되는 경우
- **오류의 원인과 성격에 따른 구분**
  - 구문 오류(syntax error, 문법 오류)
    - 프로그래밍 언어의 문법(syntax)을 잘못 기술
  - 논리 오류(logical error)
    - 내부 알고리즘이 잘못되거나 원하는 결과가 나오지 않는 등의 오류

디버깅

- **디버깅(Debugging)**
  - 프로그램 개발 과정에서 발생하는 다양한 오류를 찾아서 소스를 수정하고 해결하는 과정

  - 디버거(Debugger)
    - 디버깅을 도와주는 프로그램
    - 버그(bug): 프로그램 오류

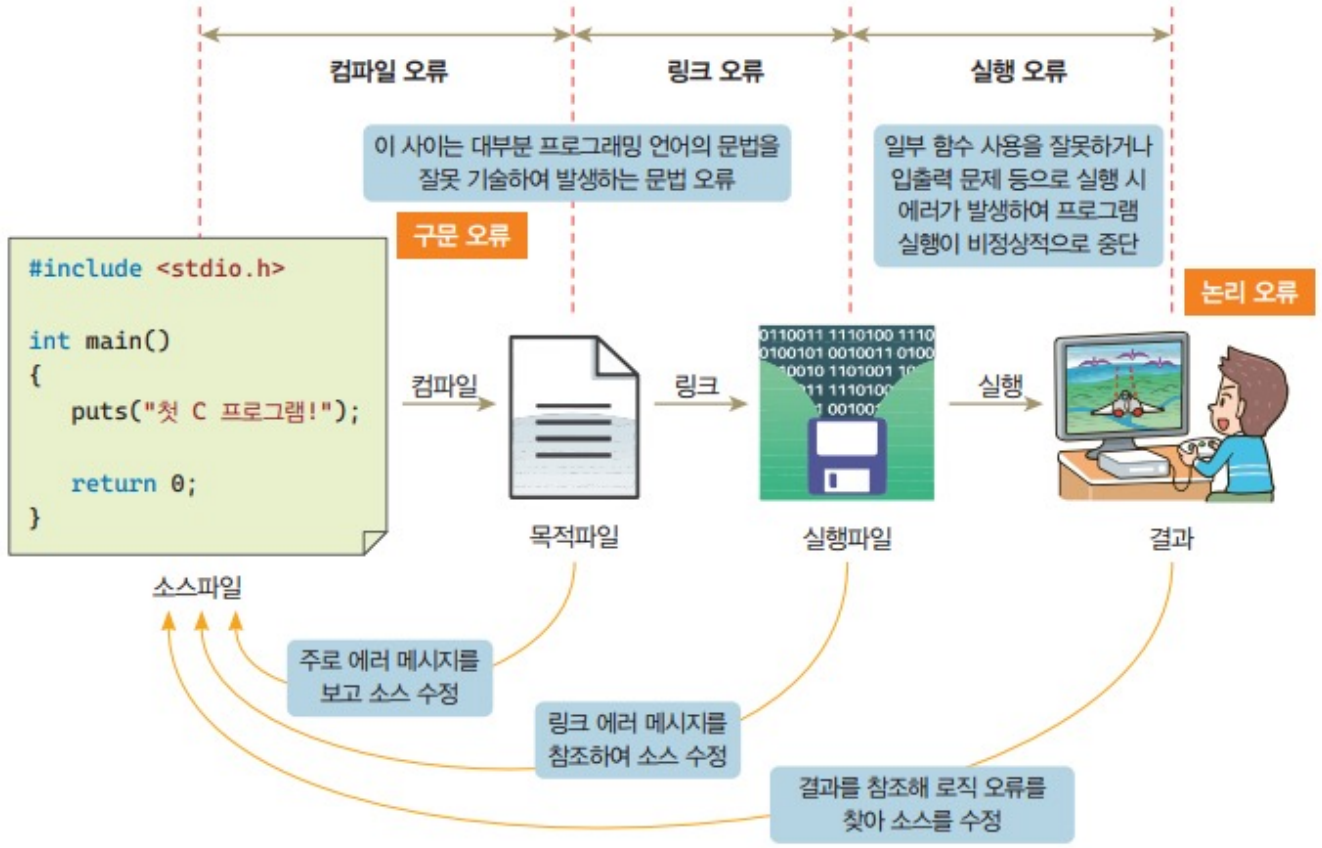


그림 2-5 디버깅의 순환 과정

프로그램 구현 과정

- **컴파일과 링크, 실행 시 오류가 발생**
  - 소스 코드를 수정해서 다시 컴파일, 링크, 실행

그림 2-9 프로그램 구현 과정 순서도

통합개발환경

- **IDE(Integrated Development Environment)**
  - 프로그램 개발에 필요한 편집기(editor), 컴파일러(compiler), 링커(linker), 디버거(debugger) 등을 통합
  - 편리하고 효율적인 통합 개발환경

그림 2-10 통합개발환경의 이해

첫 예제

- **문자열 "hello, world"가 콘솔 창에 출력되는 프로그램**
  - printf()의 괄호 사이에 기술된 "hello, world"가 출력
    - 대소문자를 구별
      - include, stdio.h, int, main, printf, return
    - 특별한 의미의 여러 문자들로 구성
      - #, <, >, (,), ;, {, }

```c
#include <stdio.h>

int main()
{
    printf("Hello World\n");
    printf("C programming\n");
    printf("안녕하세요\n");

    return 0;
}
```

```
Hello World
C programming
안녕하세요
```

소스 작성의 주의점과 들여쓰기의 이해

- **세미콜론(;)**
  - 문장의 종료를 표시
  - 콜론 :으로 잘못 입력하면 컴파일에 문제가 발생
- **들여쓰기(indentation)**
  - IDE의 편집기에서 자동으로 맞추어 주며
  - 소스의 가독성(readability)을 위해 반드시 필요

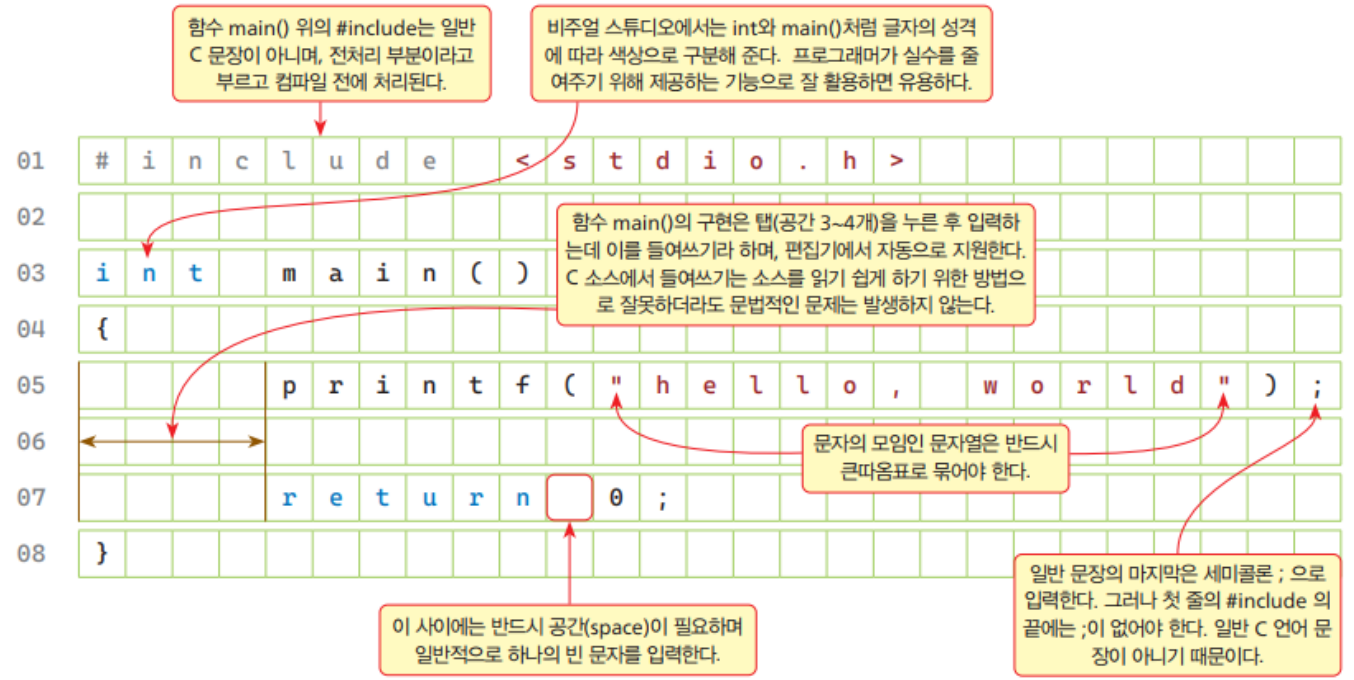


그림 2-38 소스가 기록된 원고지 모습과 주의점

헤더 파일과 프로그래밍 실행 과정

- **첫 줄에 #include <stdio.h> 삽입**
  - stdio.h 를 현재 위치에 삽입하라는 명령
    - ➢ 함수 puts()와 printf()를 사용하기 위해서 #include <stdio.h>사용

- **함수 puts("문자열"): put string**
  - 문자열을 현재 위치에 출력 후 다음 줄 첫 열로 이동
  - puts("")와 같이 공백 문자열 입력 가능
    - ➢ 현재 위치에 공백 문자열 출력 후 다음 줄로 이동

- **함수 printf()**
  - 원하는 문자열을 괄호 ("원하는 문자열") 사이에 기술
  - printf("")와 같이 공백 문자열을 인자로 전달 가능
  - printf("\n"): 한 줄 띄움
    - ➢ 다음 줄 첫 번째 컬럼의 위치로 이동

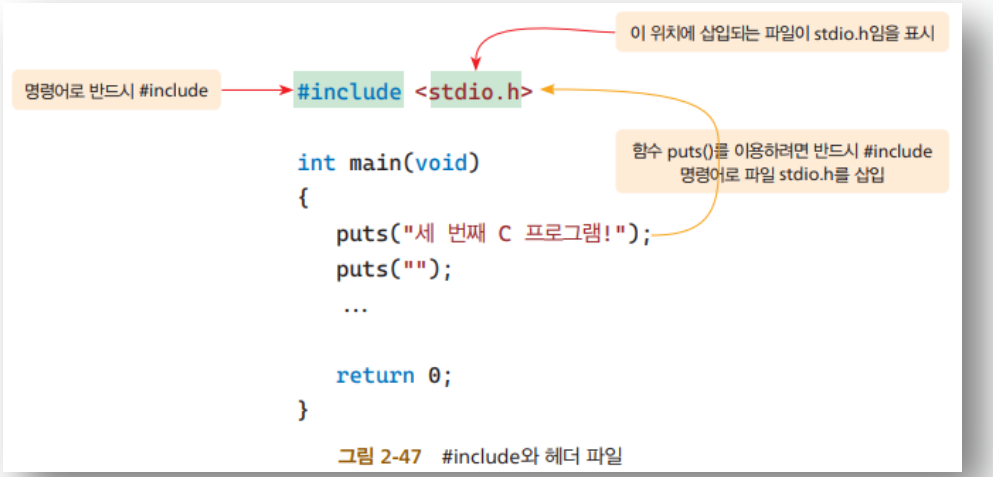


그림 2-47 #include와 헤더 파일

printf(), puts() 예제: 02put.c

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, world\n");
    puts("C 언어, 재미있다!");
    puts("화면 출력 연습");

    return 0;
}
```

"\n": 다음 줄 첫 번째 컬럼으로 이동('\r\n') 기능
- Carriage return('\r'): 첫 번째 컬럼으로 이동
- Line feed(new line)('\n'): 한 줄 띄움

```
Hello, world
C 언어, 재미있다!
화면 출력 연습
```

Hello, world \n

carriage return('\r')

line feed('\n')

C 언어, ...

main() 함수 및 함수 형태

- **C프로그램에서 main() 함수**
  - 프로그램 실행 시 가장 먼저 호출되는 특별한 함수
    - C Runtime Startup function

- **함수의 형태**
  - 함수 머리 (function header)
    - 함수 이름 및 전달 인자 선언
  - 함수 몸체(function body)
    - 실제 코드가 구현된 부분
  - 사용자 정의 함수(user defined function)
    - 프로그래머가 직접 만드는 함수
  - 라이브러리 함수(library function)
    - 시스템이 미리 만들어 놓은 함수

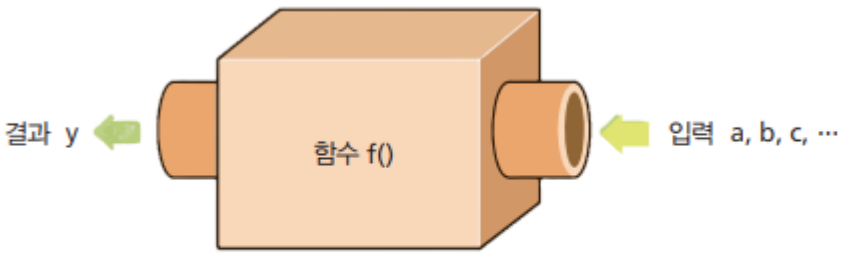
```c
#include <stdio.h>

int main(void)
{
    printf("Hello, world\n");
    puts("C 언어, 재미있다!");
    puts("화면 출력 연습");

    return 0;
}
```

함수 머리

함수 몸체



$$y = f(a, b, c, \cdots)$$

다양한 구문 오류 수정

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("C programming language.\n);
6
7      return 0;
8  }
```

닫는 따옴표가 없습니다. C/C++(8)
View Problem (⌥F8) Quick Fix... (⌘.)

마지막에 큰 따옴표가 없음
(No double quotation marks)

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("C programming language.\n");
6
7      retun 0;
8  }
```

식별자 "retun"이(가) 정의되어 있지 않습니다. C/C++(20)
View Problem (⌥F8) Quick Fix... (⌘.)

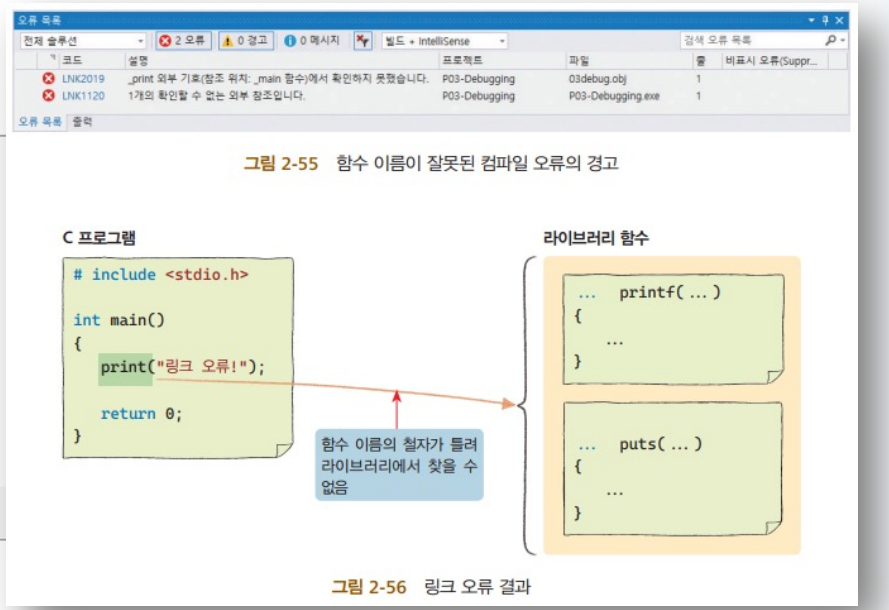
오타: return

링크 오류 수정

- **오브젝트 파일들을 실행 파일로 만드는 링크 과정에서 발생하는 오류가 '링크 오류'**
- **대표적인 링크 오류**
  - 라이브러리 함수인 printf()의 철자를 잘못 기술하는 경우
    - ➢ 마지막 f를 빼고 print()로 기술

```
1  #include <stdio.h>
2
3  int main()
4  {
5      print("C programming language.\n");
6
7      return 0;
8  }
```

오타: printf




```
error_ex.c: In function 'main':
error_ex.c:5:9: error: implicit declaration of function 'print'; did you mean 'printf'? [-Wi
mplicit-function-declaration]
    5 |         print("C programming language.\n");
      |         ^~~~~
      |         printf
```

논리 오류 수정

- **논리 오류 (logical error) 예**
  - 교과목의 성적을 산출하는 프로그램
    - 잘못된 성적 결과
- **복잡하고 큰 규모의 소프트웨어의 개발 과정에서**
  - 다양한 문제로 발생하는 논리 오류는 찾기가 매우 어려움
    - 결국 가능한 한 프로그램의 문제해결 절차인 알고리즘을 잘 만든 후
    - 이를 준수해서 소스를 코딩해야 논리 오류가 적은 프로그램을 완성

```c
# include <stdio.h>

int main()
{
   printf("무지 개같은 내 친구!");

   return 0;
}
```

그림 2-57 문자열 출력의 논리 오류 예

Questions?