

Intro to C



C의 역사

1960년대 - C 언어의 기원

- BCPL 언어 (1966년, Martin Richards): 시스템 프로그래밍을 위해 개발
- B 언어 (1969년, Ken Thompson): BCPL을 단순화하여 개발
- B 언어는 UNIX 운영체제 일부 개발에 사용됨

1970년대 - C 언어의 탄생

- 1971년: B 언어가 PDP-11 컴퓨터에서 한계를 보임
- 1972년: Dennis Ritchie가 NB(New B) → C 언어로 발전
- 1973년: UNIX 운영체제의 대부분이 C 언어로 작성됨
 - 확장성과 이식성이 향상됨

C의 역사

1980년대 - C 표준화

- 1978년: Brian Kernighan & Dennis Ritchie의 "The C Programming Language" (K&R C) 출간
- 1983년: ANSI(미국표준협회)에서 C 표준화 작업 시작
- 1989년: ANSI C(ANSI X3.159-1989, C89) 표준 확정
- 1990년: ISO(국제표준화기구)에서 C90 표준 승인

1990년대 이후 - C 언어의 발전

- 1999년: C99 표준 발표 (가변 길이 배열, 새로운 데이터 타입 추가)
- 2011년: C11 표준 발표 (멀티스레딩 지원 강화)
- 2018년: C17 표준 발표 (소규모 개선)

C 언어의 중요성

- 운영체제(UNIX, Windows, Linux) 및 임베디드 시스템의 핵심 언어
- 많은 프로그래밍 언어(C++, Java, Python 등)에 영향을 미침
- 현재까지도 강력한 성능과 효율성으로 널리 사용됨

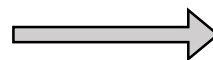
C 언어의 특징

- Middle Level Programming Language (Low-level + High-level)
 - Low-level: Direct access to memory, pointers, bitwise operations
 - High-level: Functions, structured programming
- Manual Memory Management
- Procedural Language

개발자 사고방식

문제 해결 중심

- 문제를 분석하고 해결하는 과정에 집중
- Why? -> 문제의 근본 원인을 파악
- How? -> 효율적인 해결 방법 설계



1부터 N까지의 숫자 중
하나가 빠져있는 리스트가 있다.
빠진 숫자를 찾아라.

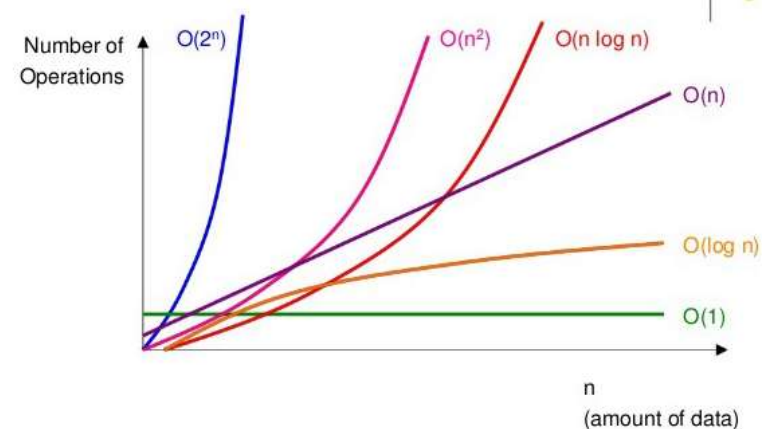
논리적 사고

- 코드의 흐름을 체계적으로 이해하고 설계

효율성 & 최적화

- 성능과 유지보수성을 고려한 코드 작성
- Big-O 개념 이해

Comparing Big O Functions



개발자 사고방식

Debugging과 Bug fix

- 버그 발생 시 원인 추적하고 수정
- 오류 메시지 분석

지속적인 학습

- 새로운 기술과 트렌드에 대한 호기심 유지
 - <https://www.youtube.com/@CppCon/videos>
 - <https://cppcon.org/>
 - <https://en.cppreference.com/w/>
 - <https://www.jendrikillner.com/tags/weekly/>
- Code Review

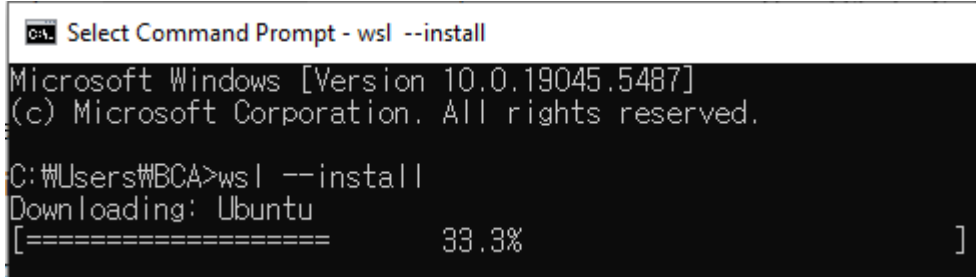
Environment Setup





WSL Install

- <https://learn.microsoft.com/en-us/windows/wsl/install>
- `wsl --install`
- `sudo apt update && sudo apt install build-essential`



```
CA: Select Command Prompt - wsl --install
Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

C:\Users\BCA>wsl --install
Downloading: Ubuntu
[===== 33.3% ]
```

WSL Install Verification

Select jinwoo@DESKTOP-UEN32NR: /mnt/c/Users/BCA

```
Microsoft Windows [Version 10.0.19045.5965]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\BCA>wsl
```

```
jinwoo@DESKTOP-UEN32NR: /mnt/c/Users/BCA$ gcc --version
```

```
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
```

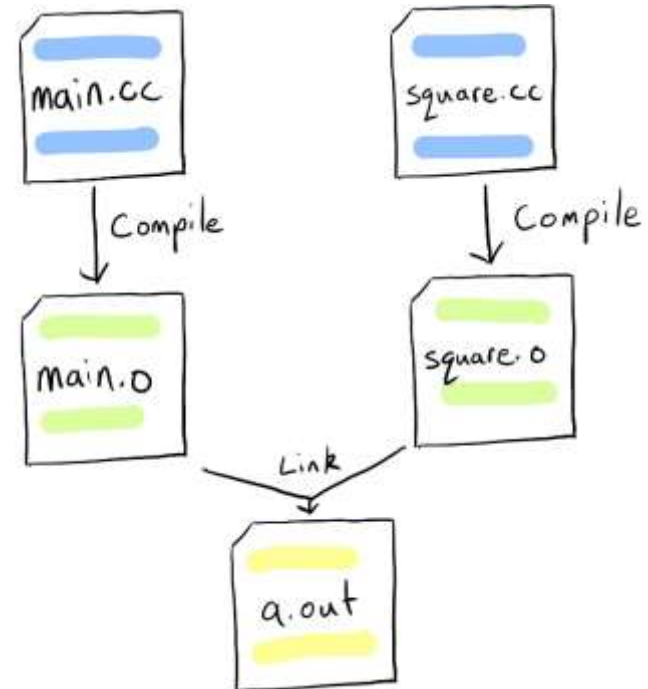
```
Copyright (C) 2023 Free Software Foundation, Inc.
```

```
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
jinwoo@DESKTOP-UEN32NR: /mnt/c/Users/BCA$
```

GCC (GNU Compiler Collection)

- Compiler?
 - > **프로그래밍 언어**로 작성된 소스 코드를 컴퓨터가 이해할 수 있는 **기계어**로 변환하는 프로그램
- gcc: GNU C Compiler
- g++: GNU C++ Compiler
 - Automatically links the std C++ libraries



GCC (GNU Compiler Collection)

```
1 | $ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -Wextra -Werror -c  
   | qdriver.c -o qdriver.o
```

```
1 | $ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -Wextra -Werror -c  
   | q.c -o q.o
```

```
1 | $ gcc q.o qdriver.o -o q.out
```

```
1 | $ ./q.out < your-in.txt > your-out.txt
```

GCC (GNU Compiler Collection)

- -std=CXX: CXX Standard를 따르도록 설정 (ex: -std=C11, -std=C99)
- -pedantic-errors: 엄격한 C 표준 적용, 표준에 맞지 않다면 Warning 대신 Error 처리
 - -pedantic은 경고 처리
- -Wstrict-prototypes: Function declaration 시 반드시 Parameter Type을 명시하도록 강제
 - int foo(); -> Warning 발생
 - int foo(void); -> Correct declaration
- -Wall: 기본적인 Warning messages 모두 활성화
- -Wextra: -Wall보다 더 많은 Warning 추가 (사용되지 않는 Variable, Parameter 등을 확인)
- -Werror: 모든 Warning을 Error로 간주 (컴파일 시 Warning 있으면 실패 처리)
- -c: Compile만 하고 Link는 하지 않음, .o (Object file)만 생성
- -o: Output file 지정 (gcc -c test.c -o result.out)

<https://gcc.gnu.org/onlinedocs/>

GCC (GNU Compiler Collection)

(hello.c)

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int a = 96;
```

```
    int b = 113;
```

```
    int c = a + b;
```

```
    printf("hello world");
```

```
    return c;
```

```
}
```

```
gcc hello.c -o hello.out
```

```
gcc hello.c
```

```
gcc -c hello.c -o hello.o
```

```
gcc hello.o -o hello.out
```