2장 프로그래밍 시작

김명호

내용

- C 언어
- 프로그래밍
- printf()
- scanf()
- 디버깅

서론

- **컴퓨터 = 하드웨어 + 소프트웨어**
- **하드웨어**
  - CPU, 메모리, 하드디스크, 메인보드, DVD-ROM, 키보드, 마우스, 모니터, . . .
- **소프트웨어**
- **운영체제**

서론

- **프로그램**
  - 컴퓨터와의 대화 수단
- **프로그래밍 환경**
  - 문서 편집기, 컴파일러
- **다양한 프로그래밍 언어**
  - 사용 목적이 다름
- **C**

C 언어

- Dennis Ritchie가 설계, 1972년 PDP-11에서 구현
- Ken Thompson와 같이 C를 사용하여 UNIX 구현

C 언어 역사

- 전통적인 C
- 1989년 ANSI C
- 1990년 ISO 표준 C
  - C90
- 1999년 개정 ISO 표준 C
  - C99

C 언어 특징

- **이식성이 좋음**
  - 대부분의 운영체제가 C로 개발됨
  - 대부분의 운영체제가 C 프로그램 환경 제공
  - 표준 라이브러리 제공
- **빠른 실행 속도**
  - 컴파일
  - 효율적인 연산자들
  - 코드 최적화

프로그래밍 준비

- **운영체제**
  - Windows 7, UNIX, LINUX
  - Cygwin(Windows 용 Linux 환경)
- **컴파일러**
  - Visual C++ (cl)
  - Borland C++ (bcc32)
  - cc
  - gcc
- **목적 파일**
  - a.out, a.exe, *.exe

프로그래밍 절차

- **프로그램 작성**
  - 문서 편집기 또는 통합환경에서는 자체 편집기
  - 파일 이름 : *filename*.c
- **컴파일**
  - gcc filename.c 또는 통합환경에서 컴파일 명령 사용
  - 컴파일 결과(목적 파일)
    - UNIX : a.out
    - DOS : *filename*.exe
- **실행**
  - UNIX : a.out
  - DOS : *filename*

C 프로그램

프로그램 1.1

```
#include <stdio.h>
int main(void)
{
    printf("나의 첫 번째 프로그램");
    return 0;
}
```

C 프로그램

- C 프로그램은 다음 빨간 부분을 반드시 가져야 함
  - 네모 부분에 실행할 코드 기술 함

```
#include <stdio.h>
int main(void)
{
    printf("나의 첫 번째 프로그램");
    return 0;
}
```

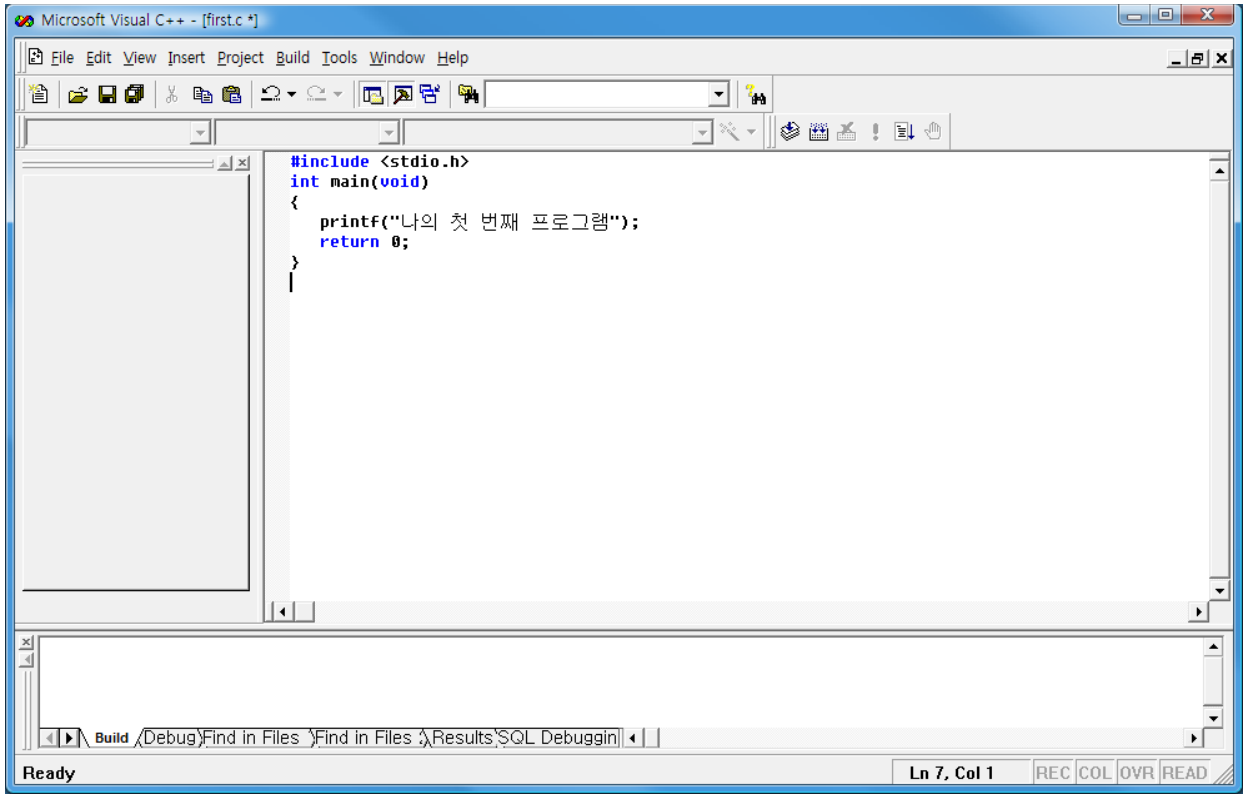
프로그래밍 예제

1. 소스 파일 만들기
   - 소스 파일 : 프로그램을 가진 파일
   - 파일 이름은 프로그램 성격에 맞는 것으로 선택하고 확장자를 .c로 함
   - first.c

프로그래밍 예제

1. 소스 파일 만들기 (통합 환경)

- Visual C++ 예



프로그래밍 예제

1. 소스 파일 만들기 (명령어 기반 환경)

- Cygwin에서 vi 사용 예

```
-tcsh
#include <stdio.h>
int main(void)
{
   printf("나의 첫 번째 프로그램");
   return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"first.c" 6L, 96C
```

프로그래밍 예제

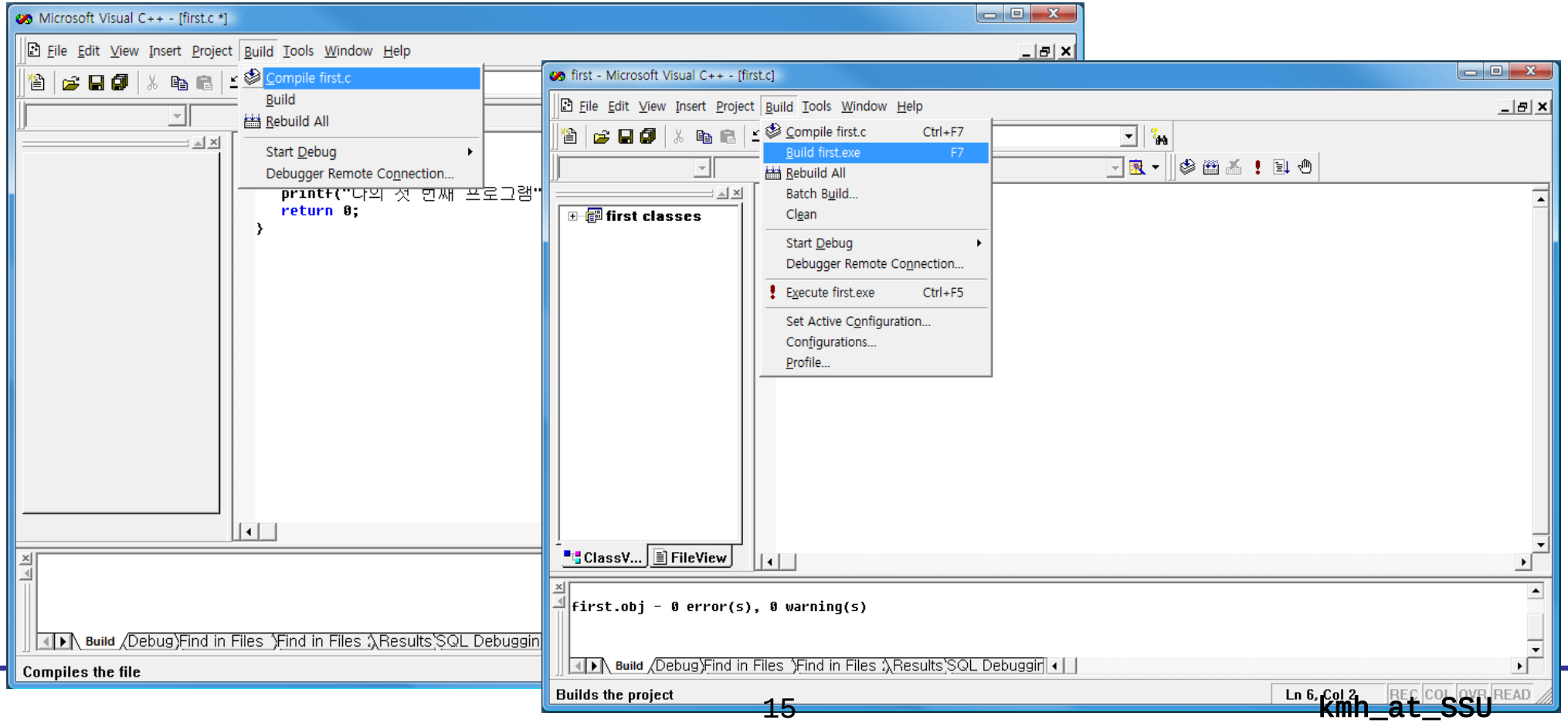
2. 컴파일

- 소스 파일은 컴퓨터가 이해할 수 없음
- 기계코드로 변환하는 것이 필요
- 컴파일러는 소스 파일을 기계코드(목적코드, 목적 파일, 실행 파일)로 변환함
  - 이러한 과정을 컴파일이라고 함
  - 실행 파일이름은 프로그래밍 환경에 따라 달라짐 : a.exe, first.exe, a.out

프로그래밍 예제

2. 컴파일 (통합 환경)

- Visual C++ 예



프로그래밍 예제

2. 컴파일 (명령어 기반 환경)

- Cygwin에서 gcc 컴파일러 사용 예

```
-tcsh
$ vi first.c
$ ls
first.c
$ gcc first.c
$ ls
a.exe  first.c
$ gcc -o first first.c
$ ls
a.exe  first.c  first.exe
$
```

프로그래밍 예제

3. 실행
   - 컴파일러가 생성한 실행 파일은 컴퓨터가 바로 실행 할 수 있음

프로그래밍 예제

3. 실행 (통합 환경)

first - Microsoft Visual C++ - [first.c]

File Edit View Insert Project Build Tools Window Help

Compile first.c Ctrl+F7
Build first.exe F7
Rebuild All
Batch Build...
Clean
Start Debug
Debugger Remote Connection...
Execute first.exe Ctrl+F5
Set Active Configuration...
Configurations...
Profile...

first classes

ClassV... FileView

first.obj - 0 error(s), 0 warning(s)

Build Debug Find in Files Find in Files Results SQL Debuggin

Executes the program

"F:₩USERS₩KMH₩C99₩Debug₩first.exe"

나의 첫 번째 프로그램Press any key to continue

프로그래밍 예제

3. 실행 (명령어 기반 환경)

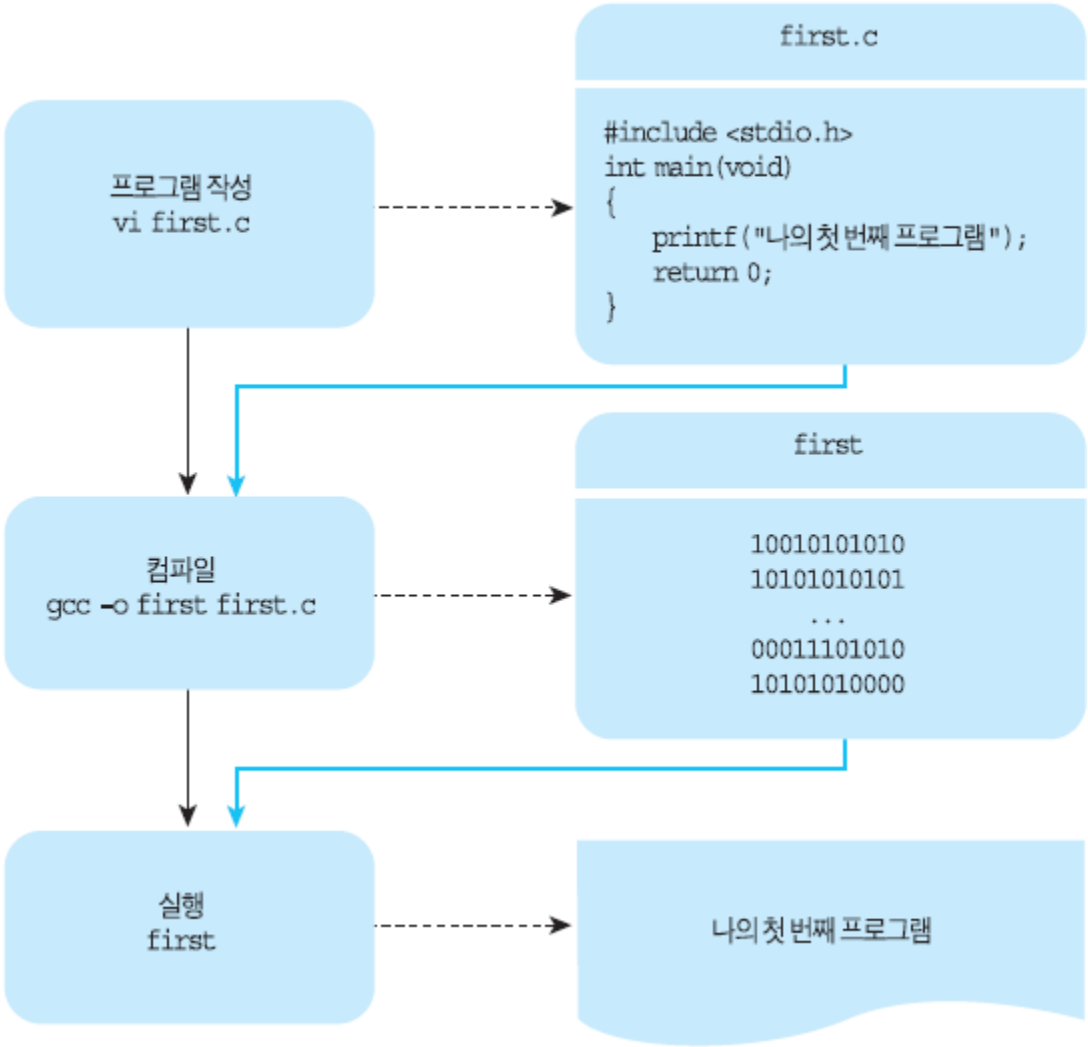
- Cygwin 예

```
-tcsh
$ vi first.c
$ ls
first.c
$ gcc first.c
$ ls
a.exe  first.c
$ gcc -o first first.c
$ ls
a.exe  first.c  first.exe
$ a
나의 첫 번째 프로그램$
$ first
나의 첫 번째 프로그램$
```

Cygwin에서의 예제

```
-tcsh
$ vi first.c
$ gcc first.c
$ ls
a.exe  first.c
$ a
나의 첫 번째 프로그램$
$ gcc -o first first.c
$ ls
a.exe  first.c  first.exe
$ first
나의 첫 번째 프로그램$
$
```

프로그래밍 절차


first.c
프로그램 작성
vi first.c
#include <stdio.h>
int main(void)
{
printf("나의첫번째 프로그램");
return 0;
}
first
컴파일
gcc -o first first.c
10010101010
10101010101
...
00011101010
10101010000
실행
first
나의첫번째 프로그램

printf()

- 화면에 출력하는 함수
- 사용 방법

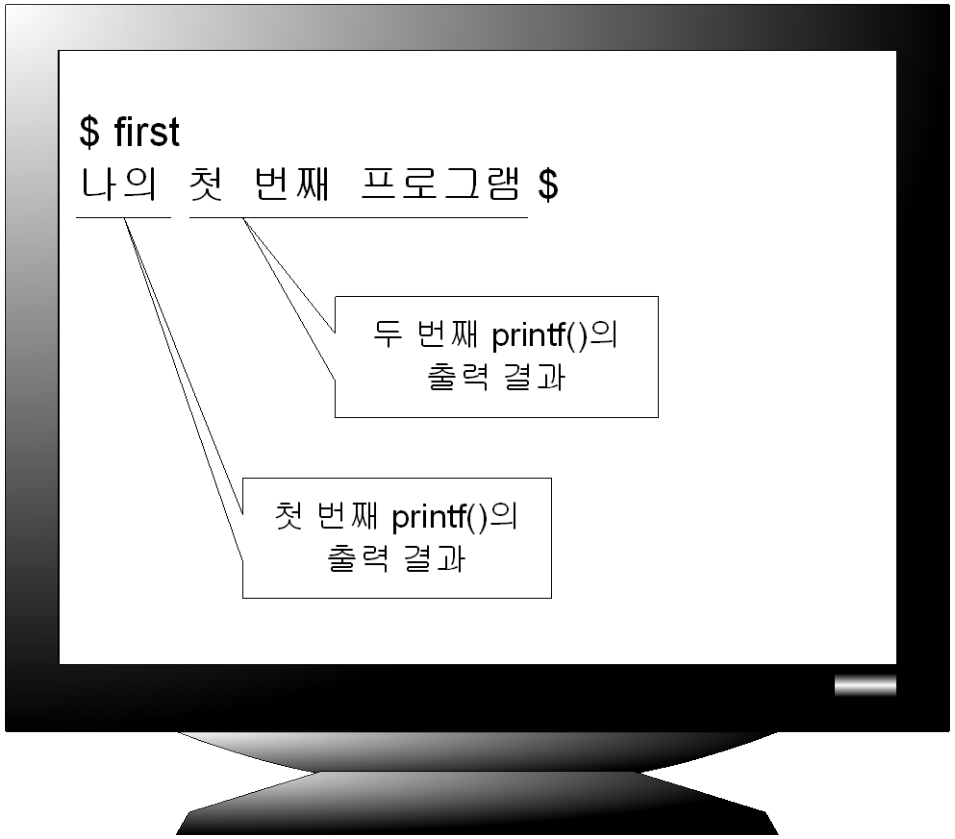
printf(“ 출력하고자 하는 내용 “);

- 연속적으로 printf()가 있을 경우, 뒤에 나오는 printf()의 출력은 바로 앞 printf()의 마지막 출력 위치에서부터 시작하여 출력한다.

printf()

프로그램 1.2

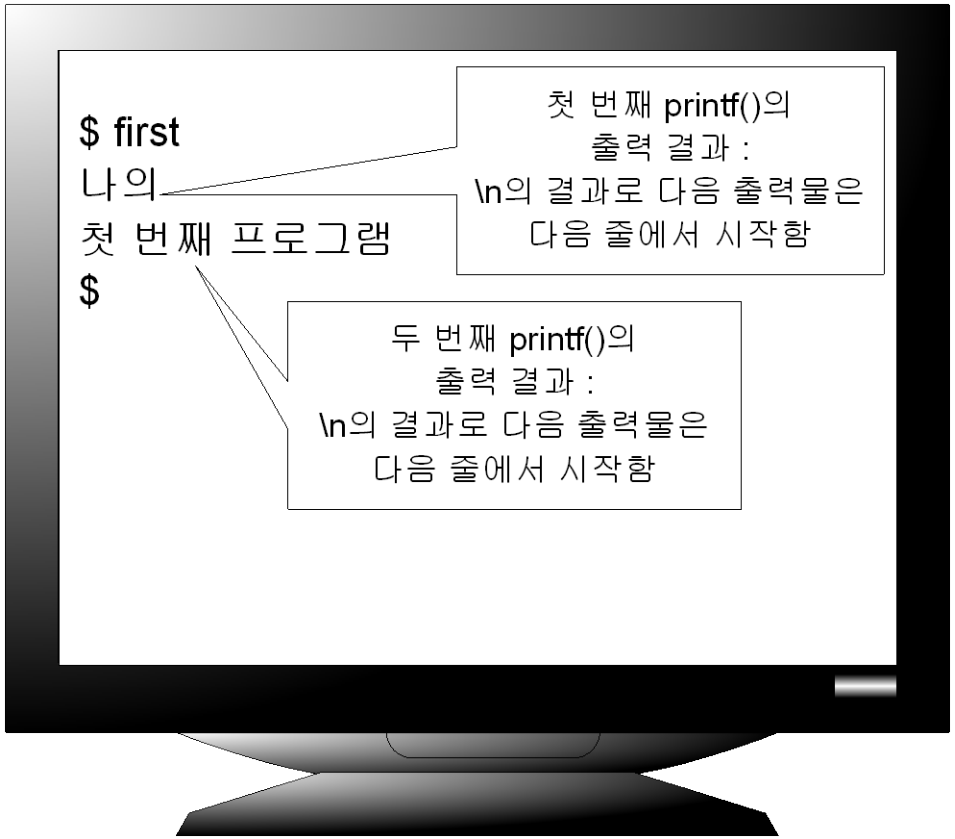
```
#include <stdio.h>
int main(void)
{
    printf("나의 ");
    printf("첫 번째 프로그램");
    return 0;
}
```



printf()

프로그램 1.3

```
#include <stdio.h>
int main(void)
{
    printf("나의\n ");
    printf("첫 번째 프로그램\n");
    return 0;
}
```



printf()

- **printf()를 이용한 계산**

```
printf("9 * 9");
printf("%d", 9 * 9);
printf("%f", 0.9 * 0.9);
```

  - **더하기 : +**

    3 + 2, 4.5 + 3.0
  - **빼기 : -**

    3 – 2, 4.5 – 3.0
  - **곱하기 : ***

    3 * 2, 4.5 * 3.0
  - **나누기 : /**

    3 / 2, 4.5 / 3.0
  - **복합연산**

    3 * 2 / 4 – 20 + 20

printf()

- **예제**

```
printf("%d", 9 * 9 + 3);
printf("답 : %d", 9 * 9 / 3);
printf("답은 %d 입니다.", 9 * 9 / 3);
```

printf()

- 인자를 여러 개 가질 수 있음
  - 인자는 괄호 안에 콤마로 분리됨
- 인자가 여러 개라도 첫 번째 인자만 출력됨
- 다른 인자들은 첫 번째 인자에게 출력 정보를 제공함
- 예

```
printf("%d * %d는(은) %d 입니다.\n", 7, 8, 7 * 8);
```

  - %로 시작하는 것을 변환 명세라고 함
  - d는 변환 문자라고 하고 대응되는 인자의 형태를 지정함
  - 각 인자는 변환 명세에 차례대로 대응 됨

```
printf("%d * %d는(은) %d 입니다.\n", 7, 8, 7 * 8);
```

printf()

프로그램 1.4

```
#include <stdio.h>
int main(void)
{
   printf("%d * %d는(은) %d 입니다.\n", 7, 8, 7 * 8);
   return 0;
}
```

프로그램 결과

```
7 * 8는(은) 56입니다.
```

printf()

프로그램 1.5

```
#include <stdio.h>
int main(void)
{
   printf("%f * %f는(은) %f 입니다.\n", 12.9, 8.8, 12.9 * 8.8);
   return 0;
}
```

프로그램 결과

```
12.900000 * 8.800000는(은) 113.520000 입니다.
```

입력 프로그램

- scanf() : 대표적인 입력 함수
- 키보드로부터 입력 받음
- printf()와 유사한 변환 문자 사용
- 사용 형식

  scanf("*입력 데이터 형*", &변수);

  - 입력 데이터 형 : %로 시작하여 변환문자로 끝남
  - 변환 명세로 지정된 형으로 입력이 안되면 입력이 안됨

scanf()

프로그램 1.6

```
#include <stdio.h>
int main(void)
{
    int year;
    printf("태어난 년도를 입력하세요 : ");
    scanf("%d", &year);
    printf("당신은 %d 살 입니다.\n", 2007 - year + 1);
    return 0;
}
```

프로그램 결과

```
$ input
태어난 년도를 입력하세요 : 1998
당신은 10 살 입니다.
$ input
태어난 년도를 입력하세요 : what
당신은 2004 살 입니다.
```

디버깅

- 컴파일러는 완벽한 프로그램만 목적 파일로 변환
- 구문 오류가 있는 프로그램을 컴파일 하면 오류 메시지가 출력됨
- 프로그램의 구문 오류가 있으면 수정해야 함
- 프로그램 오류를 수정해나가는 과정을 디버깅이라 함
- 논리 오류

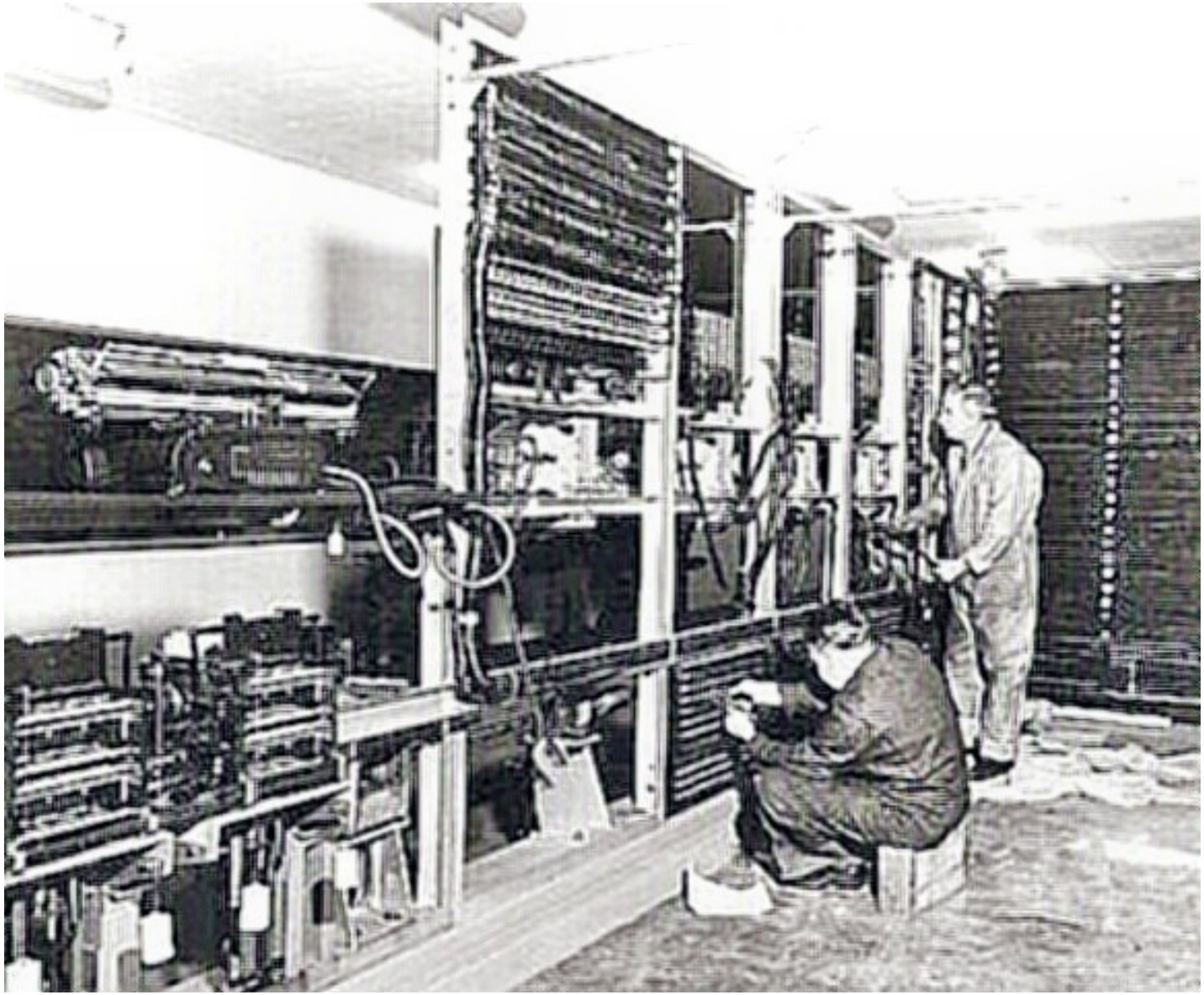
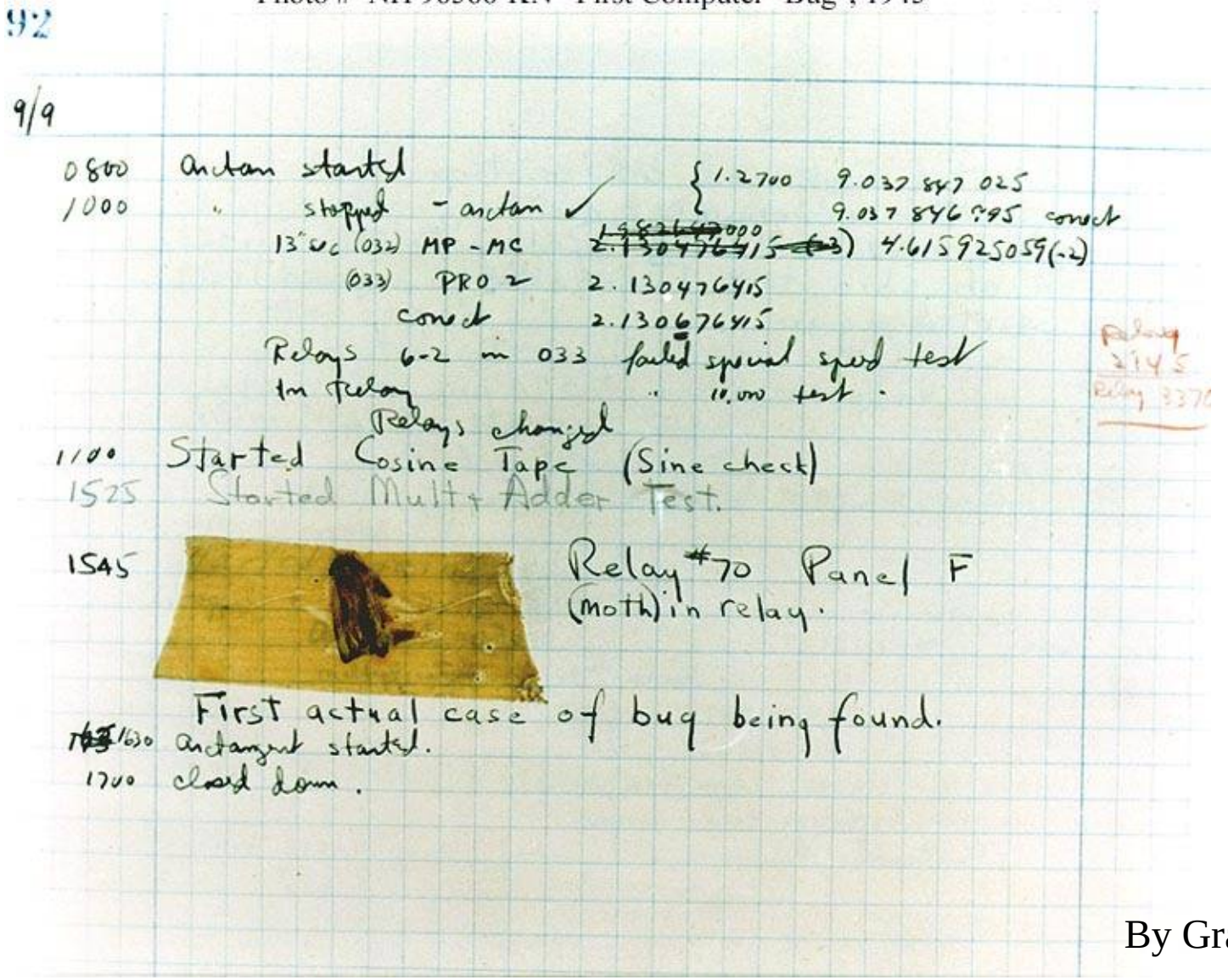


Photo # NH 96566-KN First Computer "Bug", 1945



By Grace Hopper

디버깅

프로그램 1.7

```
#include <stdio.h>
int main(void)
{
    printf("나의 첫 번째 프로그램")
    return 0;
}
```

디버깅

- **컴파일러 결과**

```
first.c: In function 'main':
first.c:5: parse error before 'return'
```

  - first.c 파일의 main 함수에서(첫 번째 메시지) first.c의 5 행의 return 문 앞에서 구문 오류가 있음(두 번째 메시지)
- **컴파일러가 출력하는 오류 메시지는 정확한 문제를 알려주지 않음**
- **연습이 필요함**

디버깅

프로그램 1.7

```
#include <stdio.h>
float main(void)
{
   printf("나의 첫 번째 프로그램");
}
```

디버깅

- 컴파일러 결과

```
first.c: In function 'main':
first.c:3: warning: return type of 'main' is not
  'int'
```

  - 경고(warning)메시지
- 경고 오류가 발생되어도 실행 파일은 생성됨
- 실행 오류가 발생할 수도 있기 때문에 경고 메시지가 나오지 않을 때까지 디버깅하는 것이 좋음