백문이불여일견,
일단 만들어보자!

Chapter 02

# 일단 짜보는 그럴듯한 C 프로그램

1장에서는 C가 뭔지 대략 파악하고 컴파일러 프로그램을 설치했다. 또한 프로그램을 어떻게 작성하고 실행하는지를 가볍게 살펴봤다. 이 장에서는 그보다 한 발 더 나아가 어느 정도 완성된 프로그램을 짜 본다.

SECTION
# 01 프로그램 작성 순서 복습

프로그램을 작성할 때에는 Visual Studio를 실행하고 Visual C++용 새 프로젝트를 생성한 후 소스 파일을 추가하는 순서로 진행하면 된다.

혼자서 C를 사용해서 제대로 된 프로그램을 만들기까지는 개인에 따라 약간의 차이가 있겠지만 최소한 한 달이나 그 이상의 시간을 투자해야 한다. 따라서 생각보다 많은 시간이 필요하므로 왠지 지루하기도 하고 부담스럽게 느껴지기도 한다.

아직 여러분 스스로가 마음의 준비를 하지 않았다는 것을 알고 있다. 하지만 축구 선수를 꿈꾸는 초등학생이 축구의 규정이나 기술을 모두 알아야 하는 것은 아니다. 우선 공을 몰고 한번쯤 무작정 운동장을 뛰어보는 것도 좋은 시작이 될 수 있지 않을까?

1장에서 우리는 100에서 50을 빼는 아주 간단한 프로그램을 작성해봤다. 이번에는 좀 그럴듯한(?) 프로그램을 만들어보자.

이 부분에서 의아하게 생각하는 사람이 있을 것이다. '이제 겨우 컴파일러 설치하고, 뭔지 모르지만 간단한 프로그램을 하나 따라해 본 것뿐인데, 벌써 프로그램을 짠다고…, 프로그램은 문법을 제대로 배운 후에 짤 수 있는 것 아닌가?' 하고 말이다.

맞는 말이다. 문법을 제대로 배운 후에야 프로그램을 짤 수 있는 기본적인 능력이 생길 것이다. 그렇지만 아직 초보에 가까운 여러분들에게 조금 그럴듯한 프로그램을 작성하도록 하는 것은 이 책을 배우면서 과연 무엇을 할 수 있는지에 대한 감을 먼저 잡아보자는 의미다. 그러므로 지금 작성하는 프로그램을 완전히 이해하지 못하더라도 잘 따라만 할 수 있으면 이 장의 목표는 충분히 달성하는 것이다.

이번에는 더하기, 빼기, 곱하기, 나누기가 실행되는 간단한 계산기 프로그램을 작성할 것이다. 다시 얘기하지만, 프로그램 문법에는 신경쓰지 말고, 전반적으로 프로그램을 작성하는 순서에 집중해서 실습해야 한다.

자, 프로그램의 작성 순서를 떠올려 보자. 이 순서대로 프로그램을 만들 것이다.

## 1 두 번째 프로젝트 만들기

1장에서 보았던 프로그램 작성 순서를 복습하면서 새로운 프로젝트를 만들어 보자. 이번에 작성할 프로젝트 이름은 'Second'로 하겠다.

**01** Visual Studio를 실행한다.

**02** 프로젝트를 생성하기 위해 메뉴의 [파일] → [새로 만들기] → [프로젝트]를 클릭한다. [새 프로젝트] 창에서 왼쪽 트리의 'Visual C++'를 선택하고 오른쪽의 'Win32 콘솔 응용 프로그램'을 선택한다. 아래쪽 이름에 'Second'를 입력한 후에 위치의 〈찾아보기〉를 클릭해서 'C:\C소스' 폴더를 선택한다. '솔루션용 디렉터리 만들기' 체크박스는 끄고 〈확인〉을 클릭하자.

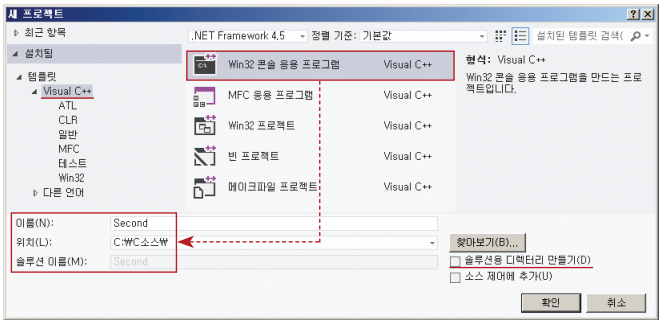



그림 2-1 새 프로젝트인 Second 생성

**03** [Win32 응용 프로그램 마법사 시작] 창에서 〈다음〉을 클릭한다.

**04** [응용 프로그램 설정] 창에서 '콘솔 응용 프로그램', '빈 프로젝트'를 체크하고 〈마침〉을 클릭한다.


그림 2-2 응용 프로그램 설정

**05** 최종적으로 다음과 같이 빈 프로젝트(또는 솔루션)를 완성했다. 이 프로젝트의 이름은 Second다.

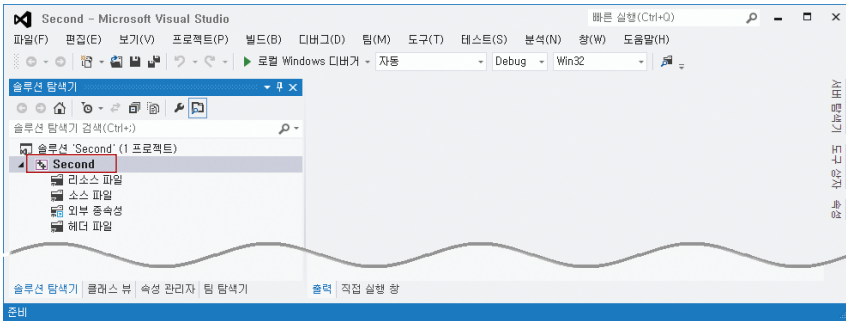



그림 2-3 생성된 Second 프로젝트

**06** 메뉴의 [파일] → [끝내기]를 선택해서 Visual Studio를 종료하자.

**SELF STUDY**

프로젝트 이름을 Test2, Test3으로 해서 새로운 프로젝트 두 개를 빠른 속도로 만들어 보자.

## 2 프로그램 코딩

**01** Visual Studio를 실행한다.

**02** Visual Studio의 메뉴에서 [파일] → [열기] → [프로젝트/솔루션]을 선택한 후 앞서 작업했던 'C:\C소스\Second' 폴더의 Second.sln을 선택한다.

**03** 왼쪽 [솔루션 탐색기]의 프로젝트 이름(지금은 Second) 아래의 '소스 파일' 폴더에서 〈마우스 오른쪽〉 버튼을 클릭한 후 [추가] → [새 항목]을 선택한다.

**04** [새 항목 추가] 창에서 'C++ 파일(.cpp)'을 선택한 상태에서 이름을 'Second.c'로 입력하고 〈추가〉를 클릭한다.

**05** 오른쪽 코드 편집 창에 100과 50의 더하기 · 빼기 · 곱하기 · 나누기를 수행하는 프로그램을 코딩하자.

**기본 2-1** 두 번째로 만드는 C 프로그램 2_1.c

```
01  #include <stdio.h>
02
```

```
03 int main( )
04 {
05   int a, b;                          ----- 계산할 두 숫자를 저장할 변수 a, b와
06   int result;                              결과를 넣을 변수 result를 선언한다.
07
08   a=100;                             ----- a에 100, b에 50을 넣는다.
09   b=50;
10
11   result = a + b ;                   ----- a와 b를 더한 결과를 result에 넣는다.
12   printf(" %d  +  %d  =  %d \n", a, b, result);   ----- 모니터에 a, b, result를 출력한다.
13
14   result = a - b ;                   ----- a와 b를 뺀 결과를 result에 넣는다.
15   printf(" %d  -  %d  =  %d \n", a, b, result);   ----- 모니터에 a, b, result를 출력한다.
16
17   result = a * b ;                   ----- a와 b를 곱한 결과를 result에 넣는다.
18   printf(" %d  *  %d  =  %d \n", a, b, result);   ----- 모니터에 a, b, result를 출력한다.
19
20   result = a / b ;                   ----- a를 b로 나눈 결과를 result에 넣는다.
21   printf(" %d  /  %d  =  %d \n", a, b, result);   ----- 모니터에 a, b, result를 출력한다.
22 }
```

**06** 틀린 글자가 없는지 확인한 후 메뉴의 [파일] → [모두 저장]을 선택해서 입력한 내용을 저장한다.

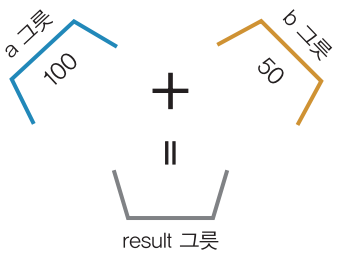
[기본 2-1]에는 배우지 않은 내용이 많이 나온다. 하나씩 살펴보면서 이해해보자.

우선 변수의 개념을 잡고 넘어가는 것이 좋을 듯하다. 변수란 간단히 '값을 저장하는 그릇(또는 방)' 정도로 생각하면 된다.

5행과 6행에서 변수(그릇) 세 개를 준비한다.


8행과 9행에서 a 그릇에는 100을, b 그릇에는 50을 넣는다.

그리고 11행에서 a 그릇의 값과 b 그릇의 값을 더한 결과를 result 그릇에 넣는다.




12행의 내용 중 printf( )는 괄호 안의 내용을 모니터에 출력하라는 의미다. printf( )는 결과를 출력하기 위해 사용하는 함수로서, 앞으로 프로그래밍을 하게 되면 수도 없이 사용하게 될 것이다. '%d'는 정수를 출력하기 위해 필요하다고 생각하면 된다. 100, 50, 150은 모두 정수이므로 그에 대응하는 '%d'를 사용한 것이다. 또 제일 뒤에 있는 '\n'은 실제로 출력되지 않는 기호로서, 다음 줄로 넘어가라는 뜻이다.

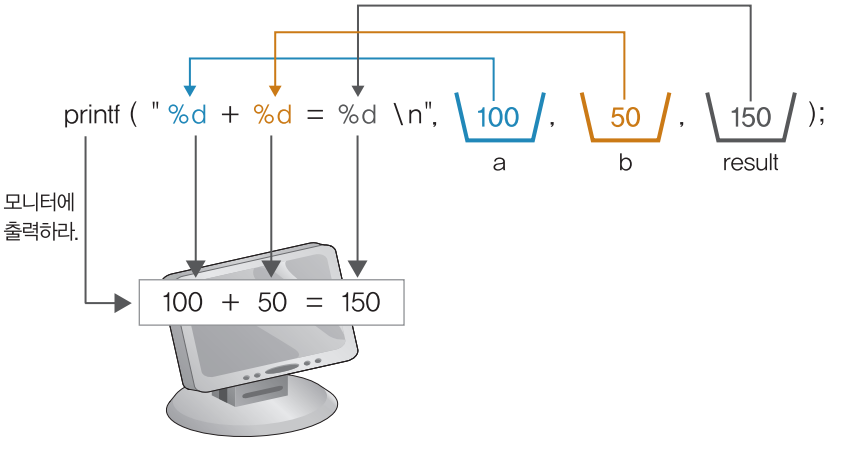



그림 2-4 printf( ) 함수의 이해

나머지 행인 14행~21행 역시 연산자만 다를 뿐 개념은 같다.

여기서 잠깐

**변수와 그릇의 차이**

변수를 그릇에 비유하는 것이 좀 유치할 수 있으나, 실제로도 별반 차이가 없다. 그리고 필자는 아직도 그렇게 이해하고 있다. 단, 진짜 그릇과 다른 점이 있다면 a 그릇과 b 그릇의 값인 100과 50은 없어지지 않고 그대로 있다는 점이다(그릇의 내용이 실제로 물이라면 없어졌겠지만 말이다).

## 3 빌드(컴파일/링크)

**01** 메뉴의 [빌드] → [솔루션 빌드]를 선택해서 프로젝트를 빌드한다.

**02** 특별히 문제가 없다면 다음 그림과 같이 '성공 1, 실패 0, 최신 0, 생략 0'이 출력된다.

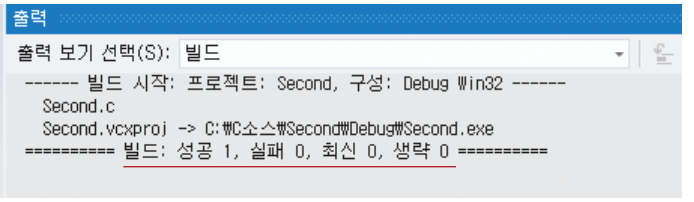


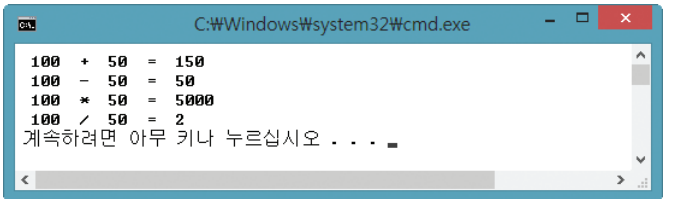

그림 2-5 프로젝트의 빌드 성공

**03** 만약, 실패가 나오면 소스에서 틀린 부분이 있다는 뜻이므로 소스에서 틀린 부분을 찾아 수정한 후 다시 빌드해야 한다.

## 4 실행

**01** Ctrl + F5를 눌러서 실행한다. 실행결과를 보면 더하기 · 빼기 · 곱하기 · 나누기의 결과가 나온 것을 확인할 수 있다.

실행결과▼




**02** 결과를 확인한 후 아무 키나 누르면 결과 창이 닫힌다.

**SELF STUDY**

[기본 2-1]을 수정하여 1234와 456의 덧셈 · 뺄셈 · 곱셈 · 나눗셈 결과를 확인해보자.

힌트_ 변수 a와 변수 b에 대입하는 값을 변경하면 된다.

**메/멘/토 퀴/즈**

❶ 화면에 무엇인가를 출력할 때 사용하는 함수는 □□□□□□( )다.

❷ a 그릇의 물을 b 그릇에 따르면 a 그릇의 물이 없어지듯이, 변수 a의 값을 변수 b에 넣으면 변수 a의 값은 없어진다. (O · X)

SECTION 02

# scanf_s( ) 맛보기

좀더 실무 프로그램에 가깝게, 사용자가 입력하는 모든 숫자가 계산되도록 프로그램을 수정해보자. scanf_s( ) 함수를 사용하면 키보드로 값을 입력받을 수 있다.

방금 전에 작성한 프로그램이 아무 이상 없이 실행되기는 했지만, 프로그램이라고 하기에는 뭔가 부족한 느낌이다. 이번에는 100과 50을 고정적으로 계산하는 것이 아니라 여러분이 직접 입력한 두 숫자의 덧셈 · 뺄셈 · 곱셈 · 나눗셈을 수행하도록 프로그램을 수정해보자.

우선 고려해야 할 부분은 [기본 2-1]의 8행과 9행의 변수 a와 변수 b의 값이다. 지금처럼 직접 소스에서 값을 변경하면 값이 바뀔 때마다 컴파일과 링크를 다시 해야 한다. 만약 이 프로그램을 판매한다고 해보자. 지금 완성된 실행 파일(Second.exe)은 100과 50에 대한 계산 결과다.

그런데 구매자가 200과 100의 계산 결과로 변경을 요청한다면 프로그램을 수정하고, 빌드한 후에 새로 생성된 실행 파일(Second.exe)을 다시 줘야 할 것이다. 즉, 새로운 값을 계산해야 할 때마다 이 과정을 반복해야 한다는 얘기다. 그러므로 소스를 수정하는 방법 대신 뭔가 다른 방법이 있을 것이고 이제 그 내용을 다룰 차례다.

## 1 값을 입력받는 scanf_s( )

변수 내용을 매번 미리 입력해두는 [기본 2-1]과 달리 실행할 때마다 키보드로 값을 입력하는 방법은 어떨까? 즉 다음과 같은 개념이다. 이렇게만 된다면 덧셈 · 뺄셈 · 곱셈 · 나눗셈을 자유자재로 수행할 수 있다.

여기서 잠깐

**scanf( )와 scanf_s( )**

C에서는 전통적으로 값을 입력받기 위해서 scanf( ) 함수를 사용해왔다. 그렇지만 최신 컴파일러는 scanf( ) 대신 scanf_s( )의 사용을 적극 권장한다. scanf( )와 scanf_s( )의 용도는 동일하지만, 사용법이 약간 달라졌다. 이는 계속해서 언급할 것이지만 한 가지 꼭 주의할 점이 있다. 바로 Visual C++ 2012 이후부터 scanf( )를 사용하면 오류가 발생해서 아예 컴파일이 안 될 수도 있다는 점이다.

참고로 scanf_s( ) 외에도 기존의 함수 이름에 '_s'가 붙은 새로운 함수는 보안이 한층 강화된 개선된 함수라고 생각하면 된다.

a에 100을 넣는다. ⇨ a에 입력할 값을 키보드로 입력받는다.
b에 50을 넣는다. ⇨ b에 입력할 값을 키보드로 입력받는다.

실행결과 ▼

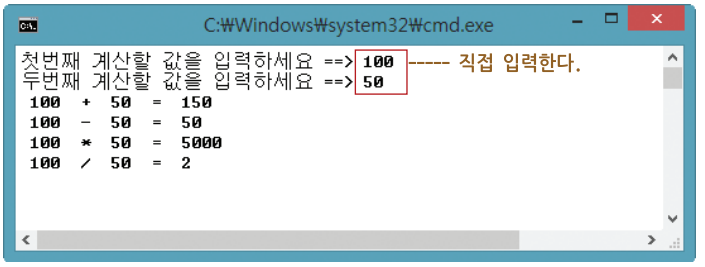



그림 2-6 사용자가 직접 값을 입력하는 프로그램

**01** 값을 입력받는 scanf_s( ) 함수를 사용해보자. [기본 2-1]의 8행과 9행을 아래 표시된 부분과 같이 수정하자.

**응용 2-2** 소스 수정하기(키보드로 값을 입력받음) 2_2.c

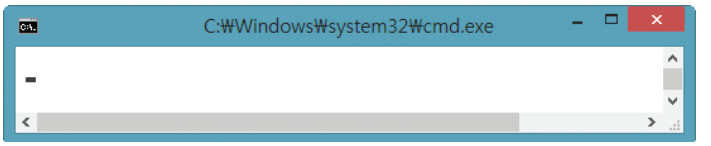
```
01 #include <stdio.h>
02
03 int main( )
04 {
05   int a, b;
06   int result;
07
08   scanf_s("%d", &a);      ----- 키보드로 a에 들어갈 값을 입력받는다.
09   scanf_s("%d", &b);      ----- 키보드로 b에 들어갈 값을 입력받는다.
10
11   result = a + b ;
12   __1__(" %d  +  %d  =  %d \n", a, b, result);
13
14   ~~~ 이하는 [기본 2-1]의 14행~21행과 동일함 ~~~
…
22 }
```

정답_ 1 printf

TIP/ 아직은 scanf_s( ) 함수에 대한 정확한 사용법을 몰라도 된다. 단지 키보드로 변수에 값을 입력하는 함수 정도로만 생각하면 된다. 그렇지만 변수 앞에는 &를 붙여야 한다는 것은 꼭 기억하자.

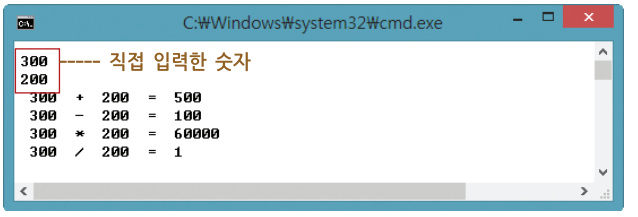
**02** Ctrl + F5를 눌러서 빌드와 실행을 동시에 진행한다. 그런데 실행결과를 보니 그냥 커서만 깜박거리는 것을 확인할 수 있다.

실행결과 ▼




**03** 일단 숫자를 하나 입력하고 Enter를 누르자. 그리고 다시 숫자 하나를 입력하고 Enter를 누르자.

실행결과 ▼




결과를 보니 첫 번째 입력한 숫자 300은 변수 a에 들어가는 값이고, 두 번째 입력한 200은 변수 b에 들어가는 값임을 알 수 있다. 즉 scanf_s("%d", &a);를 만나면 커서가 깜박거리면서 변수 a에 들어갈 값을 입력하기를 기다린다.

일단 프로그램은 문제없이 돌아가지만, 이 프로그램을 처음 사용하는 사람은 아무 설명 없이 하얀 화면만 나오므로 무엇을 해야 할지 모를 것이다.

**04** 프로그램을 좀더 편하게 사용하기 위해 다음과 같이 수정해보자.

**응용 2-3** 소스 수정하기(도움말 출력) 2_3.c

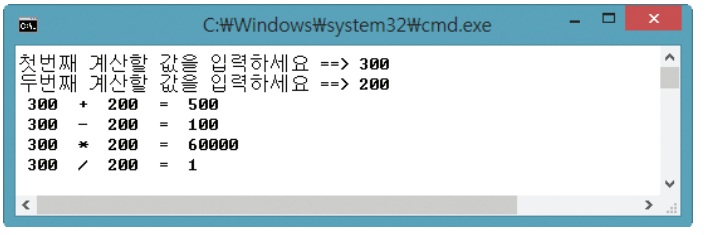
```
01 #include <stdio.h>
02
03 int main( )
04 {
05   int a, b;
06   int result;
07
08   printf("첫번째 계산할 값을 입력하세요 ==> ");      ----- 도움말을 화면에 출력한다.
09   __1__("%d", &a);
10
11   printf("두번째 계산할 값을 입력하세요 ==> ");      ----- 도움말을 화면에 출력한다.
12   __2__("%d", &b);
13
```

```
14   result = a + b ;
15   __3__(" %d  +  %d  =  %d \n",a ,b ,result);
16
17   ~~~ 이하는 [기본 2-1]의 14행~21행과 동일함 ~~~
...
25  }
```

정답_ 1 scanf_s 2 scanf_s 3 printf

**05** 다시 Ctrl+F5를 눌러서 빌드와 실행을 동시에 하자. 이제는 어떤 값이든지 입력하기만 하면 즉각 결과값이 나올 것이다.

실행결과 ▼




지금까지 조금 그럴듯한 프로그램을 작성해봤다. 물론 아직은 소스의 내용을 완전히 이해하지 못했을 것이다. 앞으로도 반복하여 설명할 것이지만, 지금까지 설명한 내용을 잘 기억해 두면 학습에 많은 도움이 될 것이다.

[응용 2-3]까지 완성한 독자라면 이제 실전에 가까운 프로그램을 작성할 준비를 마친 것이다. 다음 장부터는 본격적으로, 그리고 차근차근 C 문법을 익혀보자.

**SELF STUDY**

[응용 2-3]을 수정하여 값을 세 개 입력받아 덧셈과 곱셈을 수행해보자.

힌트_ 변수 a, 변수 b와 동일하게 새로운 변수 c를 사용한다.

**메/멘/토 퀴/즈** 키보드로 값을 입력받는 함수는 □□□□□□□( )이다. 이 함수의 괄호 안에 변수를 쓸 때는 꼭 그 앞에 □기호를 붙여줘야 한다.

여기서 잠깐

**한 번에 하는 저장, 빌드, 실행**

Ctrl+F5는 변경된 코드의 저장, 빌드, 실행을 동시에 해준다. 그러므로 소스를 수정한 후에 Ctrl+F5만 누르면 된다. 이후부터 Ctrl+F5를 누르라는 별도의 언급을 하지 않더라도 여러분이 프로그램을 작성하거나 수정한 후에는 저장, 빌드, 실행을 한 번에 진행하는 Ctrl+F5를 꼭 누르자.

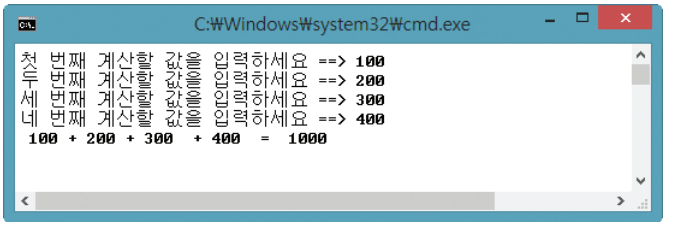
예제모음 01

# 숫자 4개를 더하는 프로그램

exam_01.c ★☆☆

**예제설명** 숫자 4개를 입력받고 그 합을 구하는 프로그램을 작성해보자.

**실행결과**



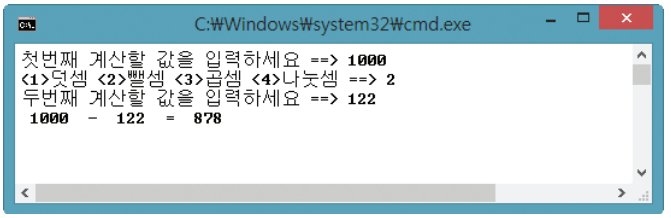

```
01 #include <stdio.h>
02
03 int main( )
04 {
05   int a, b, c, d;                          ----- 입력받을 변수 4개를 선언한다.
06   int result;
07
08   printf("첫 번째 계산할 값을 입력하세요 ==> ");
09   scanf_s("%d", &a);                       ----- 변수 a에 들어갈 값을 키보드로
10   printf("두 번째 계산할 값을 입력하세요 ==> ");                직접 입력한다.
11   scanf_s("%d", &b);                       ----- 변수 b에 들어갈 값을 키보드로
12   printf("세 번째 계산할 값을 입력하세요 ==> ");                직접 입력한다.
13   scanf_s("%d", &c)                        ----- 변수 c에 들어갈 값을 키보드로
14   printf("네 번째 계산할 값을 입력하세요 ==> ");                직접 입력한다.
15   scanf_s("%d", &d)                        ----- 변수 d에 들어갈 값을 키보드로
16                                                  직접 입력한다.
17   result = a + b + c + d;                  ----- 변수 a, b, c, d의 값을 모두 더해
18                                                  직접 입력한다.
19   printf(" %d + %d + %d  + %d  =  %d \n", a, b, c, d, result);
20 }                                          변수 a, b, c, d와 result값을 모니터에 출력한다.
```

예제모음
# 02 if문을 활용한 계산기

exam_02.c ★★☆

**예제설명** if문을 사용하여 덧셈 · 뺄셈 · 곱셈 · 나눗셈 중 하나를 선택하여 계산하는 프로그램을 작성해보자. 5장에서 배울 if문이 미리 나와서 좀 어렵게 느껴지겠지만, 직접 코딩하고 실행해보자.

**실행결과**




```
01 #include <stdio.h>
02
03 int main( )
04 {
05   int a, b;
06   int result;
07   int k;                                  ----- 계산 방식을 선택할 변수를 선언한다.
08
09   printf("첫번째 계산할 값을 입력하세요 ==> ");
10   scanf_s("%d", &a);                      ----- 계산할 두 숫자를 입력한다.
11   printf("<1>덧셈 <2>뺄셈 <3>곱셈 <4>나눗셈 ==> ");
12   scanf_s("%d", &k);                      ----- 연산자를 선택한다
13   printf("두번째 계산할 값을 입력하세요 ==> ");      (1=덧셈, 2=뺄셈, 3=곱셈, 4=나눗셈).
14   scanf_s("%d", &b);                      ----- 계산할 두 숫자를 입력한다.
15
16   if (k == 1) {                           ----- 입력한 k가 1이면 덧셈을 수행한다.
17     result = a + b ;
18     printf(" %d  +  %d  =  %d \n", a, b, result);
19   }
20
21   if (k == 2) {                           ----- 입력한 k가 2면 뺄셈을 수행한다.
22     result = a - b
23     printf(" %d  -  %d  =  %d \n", a, b, result);
24   }
25
```

```
26   if (k == 3) {                                  ----- 입력한 k가 3이면 곱셈을 수행한다.
27     result = a * b ;
28     printf(" %d  *  %d  =  %d \n", a, b, result);
29   }
30
31   if (k == 4) {                                  ----- 입력한 k가 4면 나눗셈을 수행한다.
32     result = a / b ;
33     printf(" %d  /  %d  =  %d \n", a, b, result);
34   }
35 }
```

인물로 읽는 컴퓨터 史 01

### 앨런 튜링(Alan Turing, 1911~1954)

책상 위에서 매일 접하는 현대 컴퓨터의 수학적 모델을 제시한 선지자며, "기계는 생각할 수 있는가?"라는 화두를 던져 수많은 과학자와 작가들을 고민하게 만든 천재였다. 독일군이 만든 악명 높은 에니그마(Enigma) 암호를 해킹하여 연합군이 2차 세계 대전을 승리하게 만든 숨은 영웅이었으나, 독이 든 사과를 먹고 자살했다. 컴퓨팅 분야의 노벨상이라 불리는 튜링상(Turing Prize)이 그가 세상을 뜬 지 12년 만에 만들어졌다.

#### 현대 컴퓨팅의 아버지

컴퓨터의 실체가 존재하지 않았던 20세기 초반, 대부분의 사람들은 특정한 일이나 계산을 자동으로 수행하는 기계를 만들려면 매번 그에 맞는 새로운 기계가 필요하다고 생각했다. 그리고 그것들을 동작시키려면 기계 외부의 스위치나 버튼, 선, 천공 카드 등을 조작하여 기계에게 명령을 내려야 한다고 가정하였다. 1930년대, 겨우 20대였던 앨런 튜링은 현대 컴퓨터와 프로그램이 동작하는 원리가 설명된 추상적인 수학 모델을 세상에 내놓았다. 그것이 바로 '튜링 머신(Turing Machine)'이다. 그 결과 존 폰 노이만(John von Neumann)을 비롯한 과학자들은 수많은 계산 방식을 자동으로 수행하는 디지털 컴퓨터의 핵심에 접근하게 되었다.

출처_ 누가 소프트웨어의 심장을 만들었는가(박지훈 저, 한빛미디어)

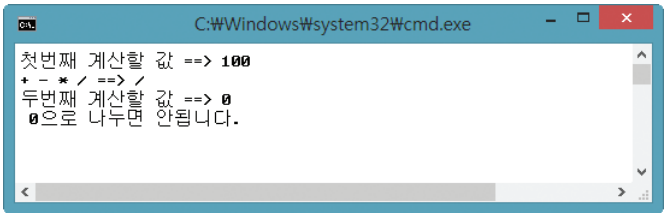
예제모음 03

# 오류가 없는 계산기

exam_03.c ★★★

**예제설명** 앞선 [예제모음 2]는 0으로 나누면 오류가 발생한다. 이것을 방지해보자. 그리고 연산자도 기호(+, −, *, /)를 사용해서 직접 입력하자. 추가로 나머지 값 연산자인 %도 계산되게 해보자.

**실행결과**




```
01  #include <stdio.h>
02
03  int main( )
04  {
05    int a, b;
06    int result;
07    char k;                    ----- 연산자를 입력받을 변수를 문자형으로 선언한다.
08
09    printf("첫번째 계산할 값 ==> ");
10    scanf_s("%d", &a);
11    printf("+ - * / % ==> ");
12    scanf_s(" %c", &k, 1);     ----- "%c"의 앞에는 공백이 있어야 한다는 것에 주의하자.
13    printf("두번째 계산할 값 ==> ");
14    scanf_s("%d", &b);
15
16    if (k == '+') {
17      result = a + b ;
18      printf(" %d  +  %d  =  %d \n", a, b, result);
19    }
20
21    if (k == '-') {
22      result = a - b ;
23      printf(" %d  -  %d  =  %d \n", a, b, result);
24    }
25
```

```
26   if (k == '*') {
27     result = a * b ;
28     printf(" %d  *  %d  =  %d \n", a, b, result);
29   }
30
31   if (k == '/') {
32     if (b != 0) {
33       result = a / b ;
34       printf(" %d  /  %d  =  %d \n", a, b, result);
35     } else
36       printf(" 0으로 나누면 안됩니다. \n");
37   }
38
39   if (k == '%') {
40     if (b != 0) {
41       result = a % b ;
42       printf(" %d  %%  %d  =  %d \n", a, b, result);
43     } else
44       printf(" 0으로 나누면 나머지 값이 안됩니다. \n");
45   }
46 }
```

----- 0으로 나누거나 나머지 값을 구하면 처리하지 않고 오류 메시지를 보여준다.

----- 0으로 나누거나 나머지 값을 구하면 처리하지 않고 오류 메시지를 보여준다.

# 요약/연습문제

## [요약]

### 1 C 프로그램 작성 순서

프로젝트 만들기 → 프로그램 코딩 → 빌드(컴파일/링크) → 실행

### 2 변수의 개념

변수는 '값을 저장하는 그릇'과 비슷한 개념이다. 그렇지만 실제 그릇과 달리 한번 들어간 값은 다른 값이 들어오기 전까지 그대로 유지된다.

### 3 scanf_s( ) 함수

키보드를 통해 값을 입력할 때 사용하는 함수로, 변수에 값을 입력받으려면 반드시 그 앞에 & 기호를 붙여야 한다.

## [연습문제]

**1 다음 소스를 보고 괄호를 채우시오.**

```
a = 500;
b = 400;
result = a + b;
```

변수 a에는 (　　)(이)가 남고, 변수 b에는 (　　)(이)가 남고, 변수 result에는 (　　)(이)가 남는다.

**2 값을 입력하는 함수는 (　　)(이)며, 변수에 값을 입력받기 위해서는 반드시 그 앞에 (　　) 기호를 붙여야 한다.**

printf( ) 함수와 상대되는 함수다.

**3 변수 a와 변수 b의 곱셈 결과를 출력하는 문장을 완성하시오.**

```
result = a * b ;
printf(" %d * %d = %d \n", __1__, __2__, __3__);
```

%d의 개수와 출력할 변수의 개수가 일치해야 한다.

**4** 다음 문장은 모니터에 무엇을 출력하는가?

```
printf("%d + %d + %d = %d", 1, 2, 3, 1+2+3);
```

**5** 다음 소스의 출력 결과를 예상하시오.

```
01 # include <stdio.h>
02
03 int main( )
04 {
05   int a, b, result;
06   a = 500;
07   b = 1000;
08
09   result = a - b;
10   printf(" %d \n", result );
11 }
```

입력하는 함수 scanf_s( )와 출력하는 함수 printf( )를 사용한다.

**6** 다음과 같이 출력되도록 프로그램을 작성하시오([예제모음 01] 참고).