

C언어 실습 교재

수원정보과학고등학교

1. C언어 시작하기

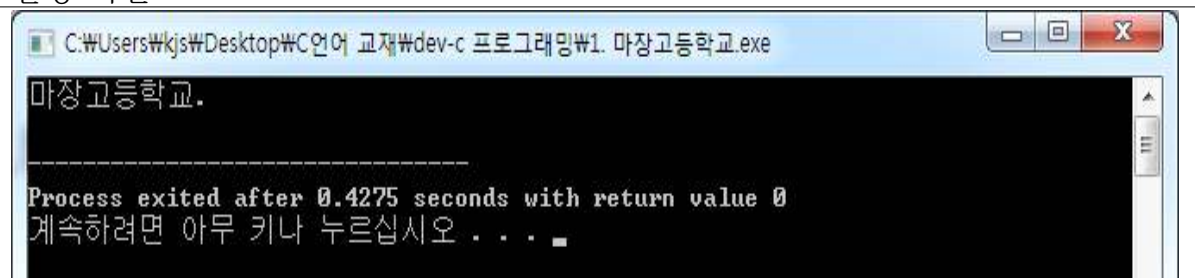
코드

```
#include <stdio.h>
//선행 처리기 Standard Input Output 헤더파일 참조

int main(void)
// main함수로 아무 인수도 전달받지 않음

{
    printf("수원정보과학고등학교.\n"); //printf함수로 수원정보과학고등학교
    라는 글자 출력
    return 0; //함수 마무리
}
```

실행 화면



//표시는 주석표시로서 프로그래밍 디버깅에 전혀 영향을 주지 않습니다.
코딩하실 때에는 //표시부터 오른쪽 문장 끝까지는 타이핑하지 마세요.

2. printf(), scanf() 함수 - 1

코드

```
#include <stdio.h>

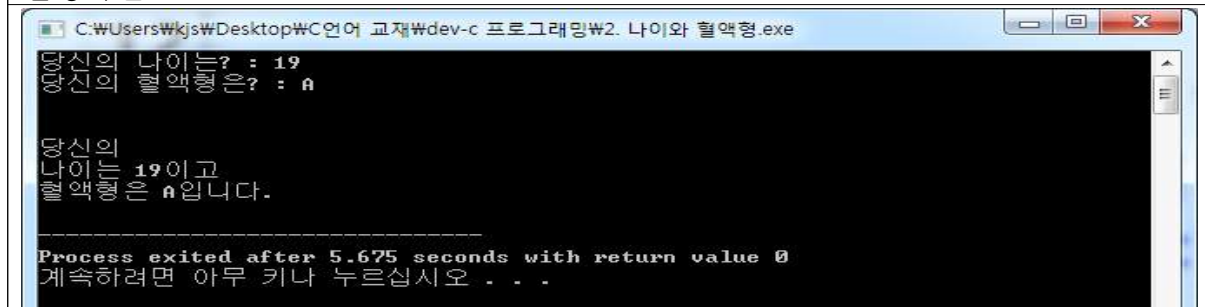
int main()
{
    int a; //정수형 변수 a 선언
    char b; //캐릭터형 변수 b 선언

    printf("당신의 나이는? : ");
    scanf("\n %d",&a);
    //정수형 데이터를 키보드로부터 받아들여서 a에 저장

    printf("당신의 혈액형은? : ");
    scanf("\n %c",&b);
    //문자형 데이터를 키보드로부터 받아들여서 b에 저장

    printf("\n\n당신의 \n");
    printf("나이는 %d이고 \n", a);
    printf("혈액형은 %c입니다. \n", b);
    return 0;
}
```

실행화면



&기호는 변수의 주소를 나타내는 포인터다. scanf()를 사용할 때 자주 사용합니다. printf()와 scanf()를 사용할 때 “”기호를 잘 사용하길 바랍니다.

3. printf(), scanf() 함수 - 2

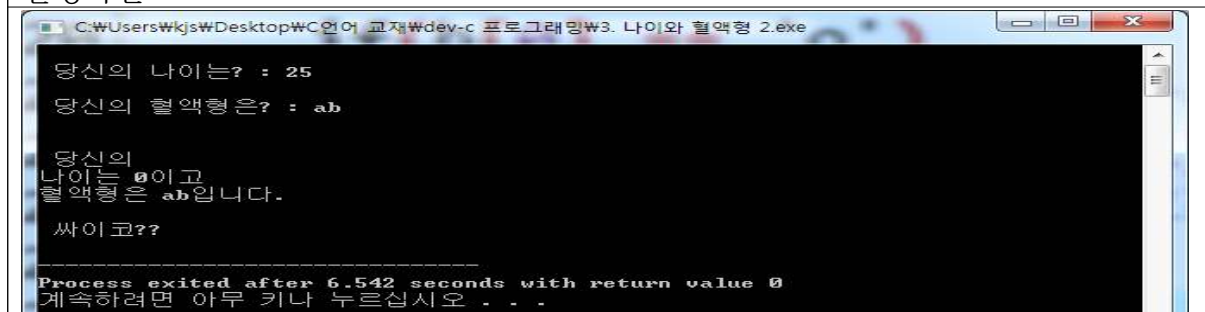
코드

```
#include <stdio.h>

int main()
{
    int a;
    char b[2];
    printf("\n 당신의 나이는? : ");
    scanf("%d", &a);
    printf("\n 당신의 혈액형은? : ");
    scanf("%s", b);
    printf("\n\n 당신의 \n");
    printf("나이는 %d이고 \n", a);
    printf("혈액형은 %s입니다. \n", b);

    if(b[0] == 'a' && b[1] != 'b')
        printf("\n 소심하시네요.\n");
    if(b[0] == 'b')
        printf("\n 이기적이지네요.\n");
    if(b[0] == 'o')
        printf("\n 활발하시네요.\n");
    if(b[0] == 'a' && b[1] == 'b')
        printf("\n AB형이시군요....\n");
    return 0;
}
```

실행화면



```
C:\Users\Wkjs\Desktop\WC언어 교재\dev-c 프로그램\3. 나이와 혈액형 2.exe
당신의 나이는? : 25
당신의 혈액형은? : ab

당신의
나이는 25이고
혈액형은 ab입니다.

싸이코??

Process exited after 6.542 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

\n 은 한 줄 띄어서 커서를 표시하는 문자로 개행문자라고 합니다.

4. 비만도 측정

코드

```
#include <stdio.h>

int main()
{
    float a, b, k = 0;

    printf("\n당신의 키는 몇 cm 입니까?");
    printf("\n키 : ");
    scanf("%f",&a);

    printf("\n당신의 몸무게는 몇 kg 입니까? : ");
    printf("\n몸무게 : ");
    scanf("%f",&b);

    k = b / ((a*0.01)*(a*0.01));
    printf("\n당신의 BMI지수는 %f입니다. \n\n",k);

    if(k<=18.5)
    {printf("당신은 저체중입니다.\n");}

    if(k>18.5 && k<=23)
    {printf("당신은 정상입니다.\n");}

    if(k>23 && k<=25)
    {printf("당신은 과체중입니다.\n");}

    if(k>25 && k<=30)
    {printf("당신은 비만입니다.\n");}

    if(k>30)
    {printf("당신은 고도비만입니다.\n");}
}
```

```

실행화면
C:\Users\Wkjs\Desktop\C언어 교재\dev-c 프로그래밍\4. 비만도 측정.exe

당신의 키는 몇 cm 입니까?
키 : 175

당신의 몸무게는 몇 kg 입니까? :
몸무게 : 75

당신의 BMI 지수는 24.489796입니다.
당신은 과체중입니다.

-----
Process exited after 3.893 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .

```

자주 사용하는 데이터 형

데이터 형	설명
int	정수형 데이터 저장 및 표시
float	소수형 데이터 저장 및 표시
char	문자형 데이터 저장 및 표시
string	문장형 데이터 저장 및 표시

위 코드에서는 비만도지수를 구하기 위해 $k = b / ((a * 0.01) * (a * 0.01))$;라는 식을 사용하게 됩니다. 식 자체에 소수점이 있으며 또한 사람의 몸무게나 신장을 나타내는 숫자는 대체로 소수점을 갖게 되므로 모든 변수를 소수점 데이터형 즉, float형을 사용한 것입니다.

if() 사용법

if(비교 및 판단문) { 명령문 }	비교 및 판단문이 참이면 '{ ' 과 '}' 사이의 명령문을 실행하고 비교 및 판단문이 거짓이면 '}' 기호 다음 명령문 실행	예시 (a > b) (a <= c) (a == b) (a != b)
-------------------------------	---	---

5. 최대값 찾기

코드

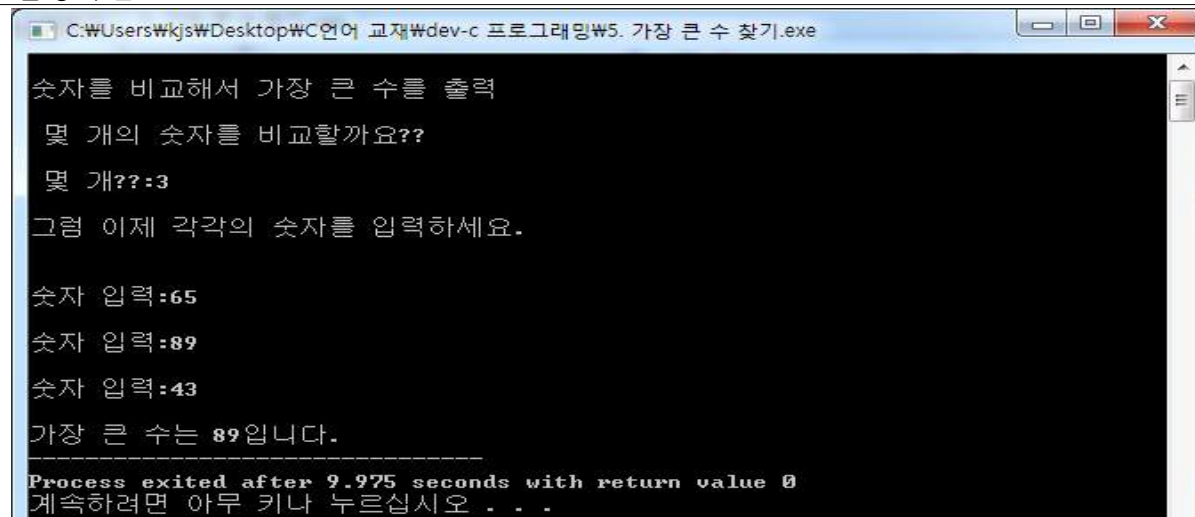
```
#include <stdio.h>

int main()
{
    int a, b, i, max = 0;

    printf("\n숫자를 비교해서 가장 큰 수를 출력");
    printf("\n\n 몇 개의 숫자를 비교할까요??");
    printf("\n\n 몇 개??");
    scanf("%d",&a);
    printf("\n그럼 이제 각각의 숫자를 입력하세요.\n\n");

    for(i=1;i<=a;i=i+1)
    {
        printf("\n숫자 입력:");
        scanf("\n%d",&b);
        if(max<b)
            max = b;
    }
    printf("\n가장 큰 수는 %d입니다.", max);
}
```

실행화면



```
C:\Users\Wkjs\Desktop\#C언어 교재\#dev-c 프로그래밍\5. 가장 큰 수 찾기.exe

숫자를 비교해서 가장 큰 수를 출력
몇 개의 숫자를 비교할까요??
몇 개?? : 3
그럼 이제 각각의 숫자를 입력하세요.

숫자 입력: 65
숫자 입력: 89
숫자 입력: 43
가장 큰 수는 89입니다.
-----
Process exited after 9.975 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

6. 최소값 찾기

코드

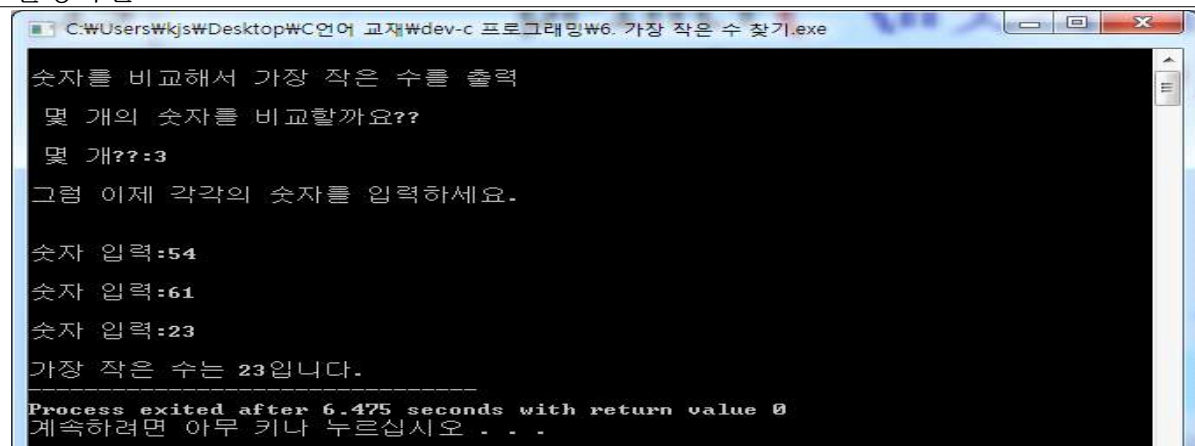
```
#include <stdio.h>

int main()
{
    int a, b, i = 0;
    int min = 1000000;

    printf("\n숫자를 비교해서 가장 작은 수를 출력");
    printf("\n\n 몇 개의 숫자를 비교할까요??");
    printf("\n\n 몇 개??");
    scanf("%d",&a);
    printf("\n그럼 이제 각각의 숫자를 입력하세요.\n\n");

    for(i=1;i<=a;i=i+1)
    {
        printf("\n숫자 입력:");
        scanf("\n%d",&b);
        if(min>b)
            min = b;
    }
    printf("\n가장 작은 수는 %d입니다.",min);
}
```

실행화면



```
C:\Users\Wkjs\Desktop\W언어 교재\dev-c 프로그래밍\6. 가장 작은 수 찾기.exe

숫자를 비교해서 가장 작은 수를 출력
몇 개의 숫자를 비교할까요??
몇 개?? : 3
그럼 이제 각각의 숫자를 입력하세요.

숫자 입력: 54
숫자 입력: 61
숫자 입력: 23
가장 작은 수는 23입니다.
-----
Process exited after 6.475 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

7. 1부터 100까지의 합 구하기

코드
<pre>#include <stdio.h> int main() { int i = 0; int sum = 0; printf("\n1부터 100까지 합하는 프로그램입니다.\n"); printf("\n자~! 1부터 100까지 더하는 과정을 보여드리겠습니다.\n"); for(i=1;i<=100;i=i+1) { sum = sum + i; printf("1부터 %d까지 더한 결과는 %d입니다.\n",i ,sum); } printf("\n1부터 100까지 합은 %d입니다.", sum); printf("\n\n끝!!"); }</pre>

printf("1부터 %d까지 더한 결과는 %d입니다.\n",i ,sum); 명령 문장은 화면에 중간결과를 출력해주는 명령입니다.

이 명령이

```
for(i=1;i<=100;i=i+1)
{
    sum = sum + i;
    printf("1부터 %d까지 더한 결과는 %d입니다.\n",i ,sum);
}
```

와 같이 반복문 안에 들어가서 중간중간 결과를 한 개씩 모두 보여주는 것입니다.

마지막 결과는 printf("\n1부터 100까지 합은 %d입니다.", sum);를 통해 확인합니다.

실행화면

```

1부터 56까지 더한 결과는 1596입니다.
1부터 57까지 더한 결과는 1653입니다.
1부터 58까지 더한 결과는 1711입니다.
1부터 59까지 더한 결과는 1770입니다.
1부터 60까지 더한 결과는 1830입니다.
1부터 61까지 더한 결과는 1891입니다.
1부터 62까지 더한 결과는 1953입니다.
1부터 63까지 더한 결과는 2016입니다.
1부터 64까지 더한 결과는 2080입니다.
1부터 65까지 더한 결과는 2145입니다.
1부터 66까지 더한 결과는 2211입니다.
1부터 67까지 더한 결과는 2278입니다.
1부터 68까지 더한 결과는 2346입니다.
1부터 69까지 더한 결과는 2415입니다.
1부터 70까지 더한 결과는 2485입니다.
1부터 71까지 더한 결과는 2556입니다.
1부터 72까지 더한 결과는 2628입니다.
1부터 73까지 더한 결과는 2701입니다.
1부터 74까지 더한 결과는 2775입니다.
1부터 75까지 더한 결과는 2850입니다.
1부터 76까지 더한 결과는 2926입니다.
1부터 77까지 더한 결과는 3003입니다.
1부터 78까지 더한 결과는 3081입니다.
1부터 79까지 더한 결과는 3160입니다.
1부터 80까지 더한 결과는 3240입니다.
1부터 81까지 더한 결과는 3321입니다.
1부터 82까지 더한 결과는 3403입니다.
1부터 83까지 더한 결과는 3486입니다.
1부터 84까지 더한 결과는 3570입니다.
1부터 85까지 더한 결과는 3655입니다.
1부터 86까지 더한 결과는 3741입니다.
1부터 87까지 더한 결과는 3828입니다.
1부터 88까지 더한 결과는 3916입니다.
1부터 89까지 더한 결과는 4005입니다.
1부터 90까지 더한 결과는 4095입니다.
1부터 91까지 더한 결과는 4186입니다.
1부터 92까지 더한 결과는 4278입니다.
1부터 93까지 더한 결과는 4371입니다.
1부터 94까지 더한 결과는 4465입니다.
1부터 95까지 더한 결과는 4560입니다.
1부터 96까지 더한 결과는 4656입니다.
1부터 97까지 더한 결과는 4753입니다.
1부터 98까지 더한 결과는 4851입니다.
1부터 99까지 더한 결과는 4950입니다.
1부터 100까지 더한 결과는 5050입니다.

1부터 100까지 합은 5050입니다.

끝!!

```

마지막 결과만 확인하는 것은 재미가 떨어질 뿐만 아니라 지금 프로그램이 잘 돌아가고 있는지 잘 모를 수 있기 때문에 이렇게 중간결과를 확인해볼 필요가 있습니다.

8. 시작 수부터 끝 수까지의 합 구하기

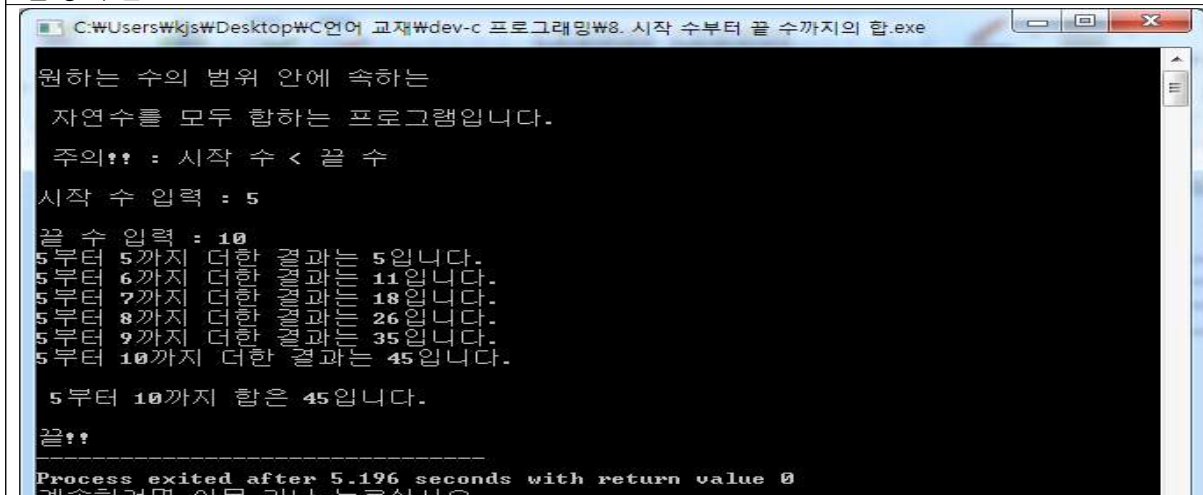
코드

```
#include <stdio.h>

int main()
{
    int a, b, i, sum = 0;
    printf("\n원하는 수의 범위 안에 속하는\n");
    printf("\n 자연수를 모두 합하는 프로그램입니다.\n");
    printf("\n 주의!! : 시작 수 < 끝 수\n");
    printf("\n시작 수 입력 : ");
    scanf("%d", &a);
    printf("\n끝 수 입력 : ");
    scanf("%d", &b);

    for(i=a;i<=b;i=i+1)
    {
        sum = sum + i;
        printf("%d부터 %d까지 더한 결과는 %d입니다.\n", a, i, sum);
    }
    printf("\n %d부터 %d까지 합은 %d입니다.", a, b, sum);
    printf("\n\n끝!!");
}
```

실행화면



```
C:\Users\Wkjs\Desktop\C언어 교재\dev-c 프로그래밍\8. 시작 수부터 끝 수까지의 합.exe

원하는 수의 범위 안에 속하는
자연수를 모두 합하는 프로그램입니다.
주의!! : 시작 수 < 끝 수
시작 수 입력 : 5
끝 수 입력 : 10
5부터 5까지 더한 결과는 5입니다.
5부터 6까지 더한 결과는 11입니다.
5부터 7까지 더한 결과는 18입니다.
5부터 8까지 더한 결과는 26입니다.
5부터 9까지 더한 결과는 35입니다.
5부터 10까지 더한 결과는 45입니다.

5부터 10까지 합은 45입니다.

끝!!
-----
Process exited after 5.196 seconds with return value 0
계속하려면 아무 키나 누르십시오
```

9. 시작 수부터 끝 수까지의 홀수 합 구하기

코드

```
#include <stdio.h>

int main()
{
    int a, b, c, i, sum = 0;

    printf("\n원하는 수의 범위 안에 속하는\n");
    printf("\n 자연수 중 모든 홀수를 합하는 프로그램입니다.\n");
    printf("\n 주의!! : 시작 수 < 끝 수\n");
    printf("\n시작 수 입력 : ");
    scanf("%d", &a);

    c = a;

    if(a%2==0)
    {c = a + 1;}

    printf("\n끝 수 입력 : ");
    scanf("%d", &b);

    for(i=c;i<=b;i=i+2)
    {
        sum = sum + i;
        printf("%d부터 %d까지 더한 결과는 %d입니다.\n", a, i, sum);
    }

    printf("\n %d부터 %d까지 중 모든 홀수의 합은 %d입니다.", a, b,
sum);

    printf("\n\n끝!!");
}
```

```
실행화면
C:\Users\Wjks\Desktop\C언어 교재\dev-c 프로그래밍\9. 시작 수부터 끝 수까지의 합(홀수).exe

원하는 수의 범위 안에 속하는
자연수 중 모든 홀수를 합하는 프로그램입니다.

주의!! : 시작 수 < 끝 수

시작 수 입력 : 3

끝 수 입력 : 16
3부터 3까지 더한 결과는 3입니다.
3부터 5까지 더한 결과는 8입니다.
3부터 7까지 더한 결과는 15입니다.
3부터 9까지 더한 결과는 24입니다.
3부터 11까지 더한 결과는 35입니다.
3부터 13까지 더한 결과는 48입니다.
3부터 15까지 더한 결과는 63입니다.

3부터 16까지 중 모든 홀수의 합은 63입니다.

끝!!
```

중간중간 결과를 확인하는 것은 더 중요한 의미가 있습니다.

본래 컴파일에 잡히는 오류는 문법오류(구문오류)일 가능성이 매우 큼니다. 즉, 논리 오류나 실행오류(런타임오류) 등과 같은 오류는 프로그래머가 직접 해결해야하는 경우가 더 많다고 볼 수 있습니다.

따라서 이럴 때 printf()를 사용하여 해결할 수 있습니다.

디버깅을 할 때 자신의 코드가 어디까지 실행되는 것인지 확인할 때 필요합니다.

중간중간에 printf("여기까지는 실행\n"); 이라는 문장을 삽입한다던가 반복문 안에서 몇 번 돌았는지 확인하기 위해 카운팅을 하는 변수를 삽입해서 출력하는 꼼수(?)를 부린다면 코딩할 때 디버깅이 더 잘 될 수 있습니다.

초보일 때부터 결과만 확인하는 코딩보다는 중간 연산의 과정을 한 개씩 확인하려는 코딩의 노력이 필요합니다.

10. 시작 수부터 끝 수까지의 짝수 합 구하기

코드

```
#include <stdio.h>

int main()
{
    int a, b, c, i, sum = 0;

    printf("\n원하는 수의 범위 안에 속하는\n");
    printf("\n 자연수 중 모든 짝수를 합하는 프로그램입니다.\n");
    printf("\n 주의!! : 시작 수 < 끝 수\n");
    printf("\n시작 수 입력 : ");
    scanf("%d", &a);

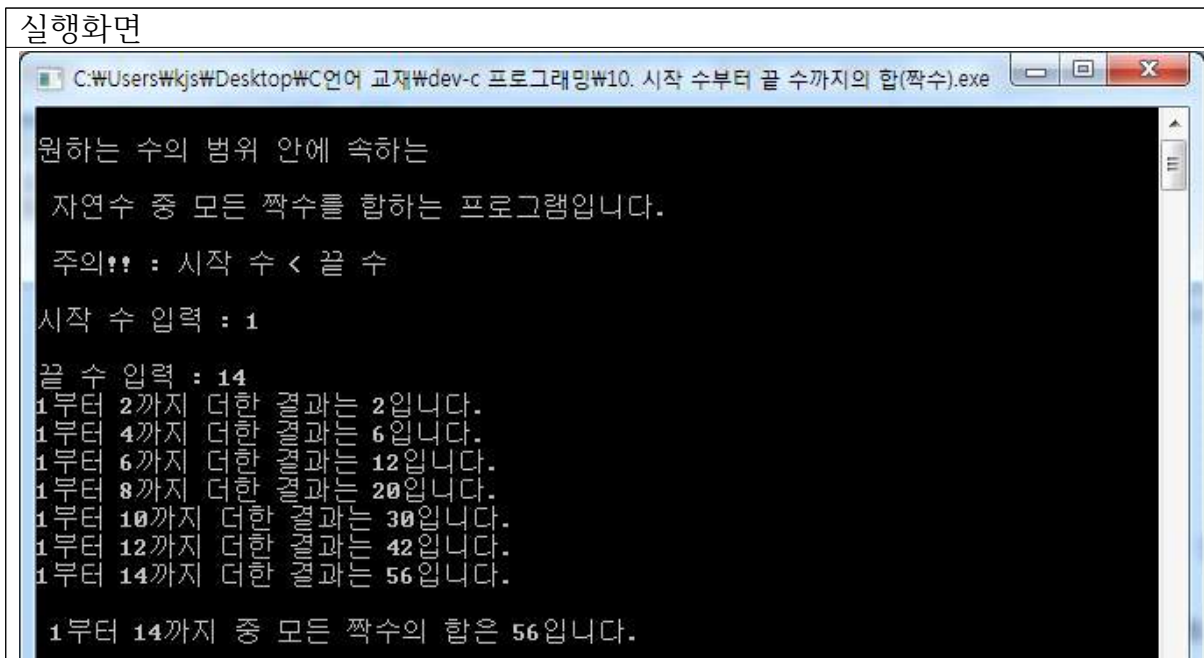
    c = a;

    if(a%2==1)
    {c = a + 1;}

    printf("\n끝 수 입력 : ");
    scanf("%d", &b);

    for(i=c; i<=b; i=i+2)
    {
        sum = sum + i;
        printf("%d부터 %d까지 더한 결과는 %d입니다.\n", a, i, sum);
    }
    printf("\n %d부터 %d까지 중 모든 짝수의 합은 %d입니다.", a, b,
sum);

    printf("\n\n끝!!");
}
```



반복문을 사용할 때에는 while, do while, for를 사용하게 됩니다.
특히 위 세 가지 함수의 사용을 익히셔야 합니다.

for(i = 1; i <= 10; i++) { 명령문 }	i를 1로 만들고 i가 100까지 1씩 증가하면서 (i <= 100)이 성립되는 동안 '{' 과 '}' 사이의 명령문을 반복 (i <= 100)이 성립되지 않으면 '}' 기호 다음 명령문 실행
---	--

컴파일 오류에 대한 종류와 설명을 해드리겠습니다.

출력 출력 보기 선택(S): 빌드 1>----- 빌드 시작: 프로젝트: 112, 구성: Debug Win32 ----- 1> 건너뛰고 있습니다... (관련된 변경 내용 없음) 1> 12.cpp ===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====	에러나 경고가 없음 즉, 문법적 오류가 발견되지 않음 실행 단계로 아무 문제 없이 이어짐
출력 출력 보기 선택(S): 빌드 1>----- 빌드 시작: 프로젝트: 112, 구성: Debug Win32 ----- 1> 12.cpp 1> error C2144: 구문 오류 : int'은(는) ';' 다음에 와야 합니다. ===== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 =====	오류가 발생됨 int 선언 전 명령어 문장에 ; 기호가 누락 int 선언 전 명령어 문장에 ; 삽입해서 해결
출력 출력 보기 선택(S): 빌드 1> error C2065: 'max' : 선언되지 않은 식별자입니다. ===== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 =====	오류가 발생됨 max 변수가 선언되지 않은 상태에서 사용됨 max 변수를 사용하기 전에 max 변수를 선언해서 해결

11. 배수 찾기

코드

```
#include <stdio.h>

int main()
{
    int a, b, c = 0;
    int i = 0;
    int cnt = 0;

    printf("\n원하는 수의 범위 안에서 \n");
    printf("\n 배수를 찾아 출력하는 프로그램입니다.\n");
    printf("\n 주의!! : 시작 수 < 끝 수\n");
    printf("\n시작 수 입력 : ");
    scanf("%d", &a);
    printf("\n끝 수 입력 : ");
    scanf("%d", &b);
    printf("\n 자. 이제 배수를 입력해주세요.\n");
    printf("\n배수 입력 : ");
    scanf("%d", &c);

    printf("%d부터 %d까지 중\n", a, b);
    for(i=a;i<=b;i=i+1)
    {
        if((i%c)==0)
        {
            cnt = cnt + 1;
            printf("%d번째 %d의 배수는 %d입니다.\n", cnt, c, i);
        }
    }
    printf("\n %d부터 %d까지 %d의 배수는 총 %d개 입니다.", a, b, c,
cnt);
    printf("\n\n끝!!");
}
```

```
실행화면
C:\Users\Wjks\Desktop\C언어 교재\dev-c 프로그래밍\11. 배수 찾기.exe

원하는 수의 범위 안에서
배수를 찾아 출력하는 프로그램입니다.
주의!! : 시작 수 < 끝 수
시작 수 입력 : 25
끝 수 입력 : 75
자. 이제 배수를 입력해주세요.
배수 입력 : 4
25부터 75까지 중
1번째 4의 배수는 28입니다.
2번째 4의 배수는 32입니다.
3번째 4의 배수는 36입니다.
4번째 4의 배수는 40입니다.
5번째 4의 배수는 44입니다.
6번째 4의 배수는 48입니다.
7번째 4의 배수는 52입니다.
8번째 4의 배수는 56입니다.
9번째 4의 배수는 60입니다.
10번째 4의 배수는 64입니다.
11번째 4의 배수는 68입니다.
12번째 4의 배수는 72입니다.

25부터 75까지 4의 배수는 총 12개 입니다.

끝!!

-----
Process exited after 8.713 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

아래의 코드를 잘 확인해주세요.

```
if((i%c)==0)
{
    cnt = cnt + 1;
    printf("%d번째 %d의 배수는 %d입니다.\n", cnt, c, i);
}
```

%연산자는 나머지를 구하는 연산자로, 배수나 약수, 소수를 구할 때 자주 사용되는 연산자입니다.

12. 공배수 찾기

코드 - 앞 부분

```
#include <stdio.h>

int main()
{
    int a, b, c, d = 0;
    int b = 0;
    int c = 0;
    int d = 0;
    int i = 0;
    int cnt = 0;

    printf("\n 원하는 수의 범위 안에서 \n");
    printf("\n 공배수를 찾아 출력하는 프로그램입니다.\n");
    printf("\n 주의!! : 시작 수 < 끝 수\n");

    printf("\n 시작 수 입력 : ");
    scanf("%d", &a);
    printf("\n 끝 수 입력 : ");
    scanf("%d", &b);

    printf("\n 자. 이제 첫번째 배수를 입력해주세요.\n");
    printf("\n 첫번째 배수 입력 : ");
    scanf("%d", &c);

    printf("\n 자. 이제 두번째 배수를 입력해주세요.\n");
    printf("\n 두번째 배수 입력 : ");
    scanf("%d", &d);
```

코드 - 뒷 부분

```
printf("%d부터 %d까지 중\n", a, b);
for(i=a;i<=b;i=i+1)
{
    if((i%c)==0)
    {
        if((i%d)==0)
        {
            cnt = cnt + 1;
            printf("%d번째 %d와 %d의 공배수는 %d입니다.\n", cnt, c, i);
        }
    }
}

printf("\n %d부터 %d까지 %d와 %d의 공배수는 총 %d개 입니다.", a,
b, c, d, cnt);
printf("\n\n끝!!");
}
```

실행화면

```
C:\Users\Wkjs\Desktop\WC언어 교재\dev-c 프로그래밍\12. 공배수 찾기.exe

원하는 수의 범위 안에서
공배수를 찾아 출력하는 프로그램입니다.
주의!! : 시작 수 < 끝 수
시작 수 입력 : 15
끝 수 입력 : 65
자. 이제 첫번째 배수를 입력해주세요.
첫번째 배수 입력 : 2
자. 이제 두번째 배수를 입력해주세요.
두번째 배수 입력 : 3
15부터 65까지 중
1번째 2와 3의 공배수는 18입니다.
2번째 2와 3의 공배수는 24입니다.
3번째 2와 3의 공배수는 30입니다.
4번째 2와 3의 공배수는 36입니다.
5번째 2와 3의 공배수는 42입니다.
6번째 2와 3의 공배수는 48입니다.
7번째 2와 3의 공배수는 54입니다.
8번째 2와 3의 공배수는 60입니다.

15부터 65까지 2와 3의 공배수는 총 8개 입니다.

끝!!

-----
Process exited after 7.195 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

13. 총점과 평균 구하기

코드

```
#include <stdio.h>

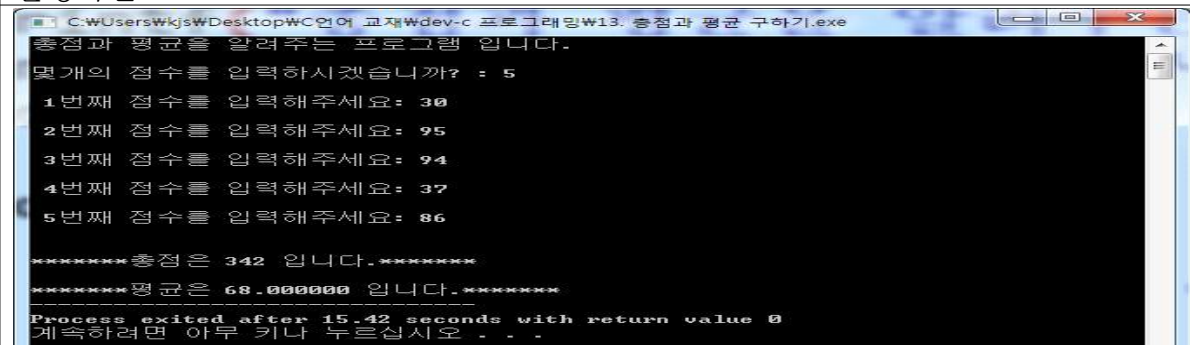
int main()
{
    int n = 0;
    int score = 0;
    int i = 0;
    int sum = 0;
    float avg = 0;

    printf("총점과 평균을 알려주는 프로그램 입니다.\n\n");
    printf("몇개의 점수를 입력하시겠습니까? : ");
    scanf("%d",&n);

    for(i=1; i<=n; i++)
    {
        printf("\n %d번째 점수를 입력해주세요: ", i);
        scanf("%d",&score);
        sum = sum + score;
    }

    avg = sum / n;
    printf("\n\n*****총점은 %d 입니다.*****",sum);
    printf("\n\n*****평균은 %f 입니다.*****",avg);
}
```

실행화면



```
C:\Users\wkjs\Desktop\WC언어 교재\Wdev-c 프로그래밍\13. 총점과 평균 구하기.exe
총점과 평균을 알려주는 프로그램 입니다.
몇개의 점수를 입력하시겠습니까? : 5
1번째 점수를 입력해주세요: 30
2번째 점수를 입력해주세요: 95
3번째 점수를 입력해주세요: 94
4번째 점수를 입력해주세요: 37
5번째 점수를 입력해주세요: 86

*****총점은 342 입니다.*****
*****평균은 68.400000 입니다.*****
Process exited after 15.42 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

14. 구구단 구하기

코드

```
#include <stdio.h>

int main ()
{
    int a = 0;
    int i = 0;
    int b = 0;

    for(;;)
    {
        printf("\n\n구구단 프로그램입니다\n");
        printf("\n몇 단을 출력하시겠습니까? : ");
        scanf("%d" , &a);
        printf("\n\n");

        for(i=1; i<=9; i++)
        {
            printf(" %d * %d = %d\n",a,i,a*i);
        }

        printf("\n\n다시 1 종료 2\n") ;
        scanf("%d" , &b);

        if(b!=1)
        {
            break;
        }
    }
}
```

실행화면

```
C:\Users\Wkjs\Desktop\2016년 업무\상업정보능력경진반 운영\C언어 수업\김태희\구구단 연...

구구단 프로그램입니다
몇 단을 출력하시겠습니까? : 2

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

다시 1 종료 2
1

구구단 프로그램입니다
몇 단을 출력하시겠습니까? : 9

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81

다시 1 종료 2
1

구구단 프로그램입니다
몇 단을 출력하시겠습니까? : 5

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45

다시 1 종료 2
```

15. 약수 구하기

코드

```
#include <stdio.h>

int main()
{
    int a = 0;
    int i = 0;
    int cnt = 0;

    printf("\n 숫자를 입력받아서 약수 출력");
    printf("\n\n 어떤 숫자의 약수를 보여드릴까요??");
    printf("\n\n 숫자 : ");
    scanf("%d",&a);

    printf("\n%d의 약수는 : \t", a);

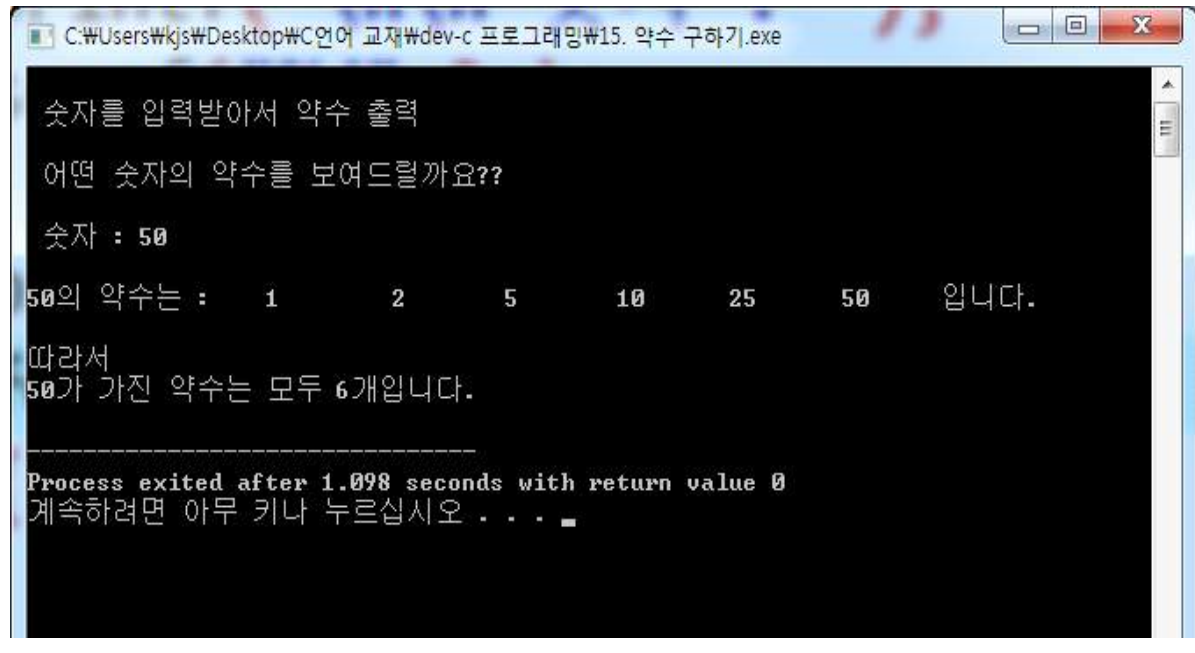
    for(i=1; i<=a; i=i+1)
    {
        if(a%i == 0)
        {
            printf(" %d\t ", i);
            cnt = cnt +1;
        }

    }

    printf("\n입니다. \n");
    printf("\n따라서");

    printf("\n%d가 가진 약수는 모두 %d개입니다.\n" ,a, cnt);
}
```

실행화면-1

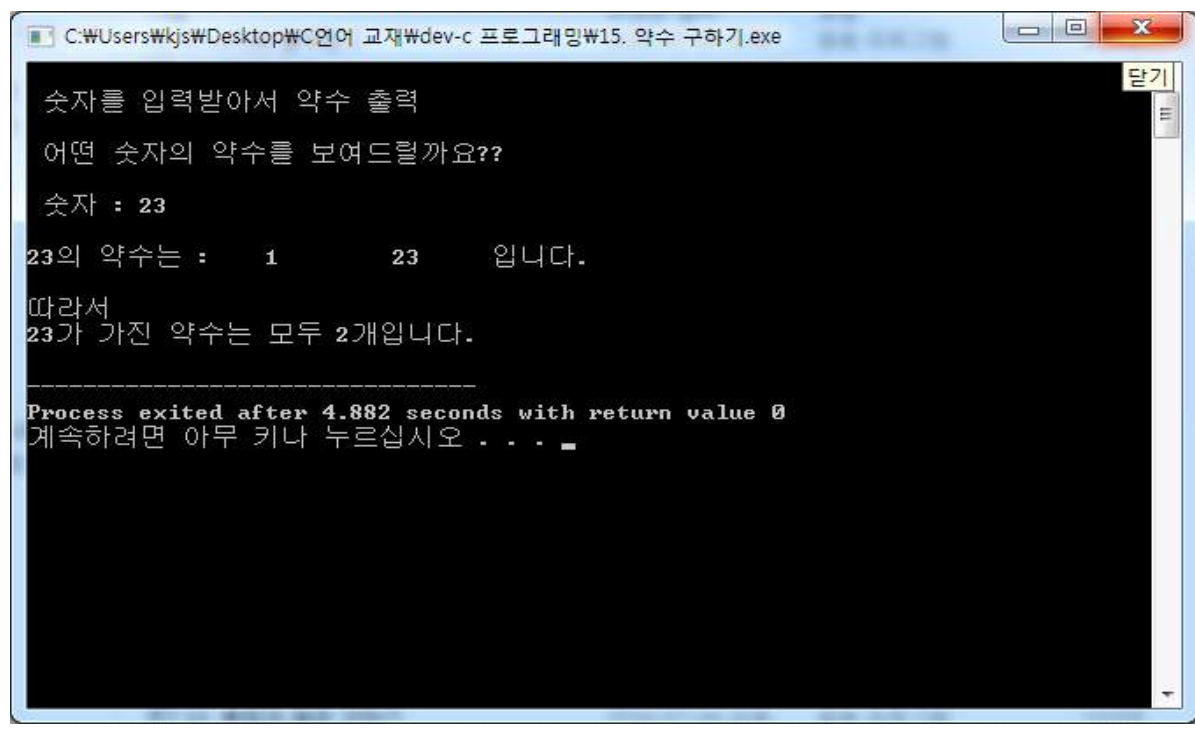


C:\Users\Wkjs\Desktop\C언어 교재\dev-c 프로그래밍\15. 약수 구하기.exe

```
숫자를 입력받아서 약수 출력
어떤 숫자의 약수를 보여드릴까요??
숫자 : 50
50의 약수는 :   1       2       5       10      25      50      입니다.
따라서
50가 가진 약수는 모두 6개입니다.

-----
Process exited after 1.098 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

실행화면-2



C:\Users\Wkjs\Desktop\C언어 교재\dev-c 프로그래밍\15. 약수 구하기.exe

```
숫자를 입력받아서 약수 출력
어떤 숫자의 약수를 보여드릴까요??
숫자 : 23
23의 약수는 :   1       23      입니다.
따라서
23가 가진 약수는 모두 2개입니다.

-----
Process exited after 4.882 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

16. 소수 구하기

코드-앞부분

```
#include <stdio.h>

int main()
{
    int a = 0;
    int i = 0;
    int cnt = 0;

    printf("\n 숫자를 입력받아서 소수인지 아닌지를 판별해드립니다.");
    printf("\n\n 어떤 숫자를 판별해드릴까요??");
    printf("\n\n 숫자 : ");
    scanf("%d",&a);

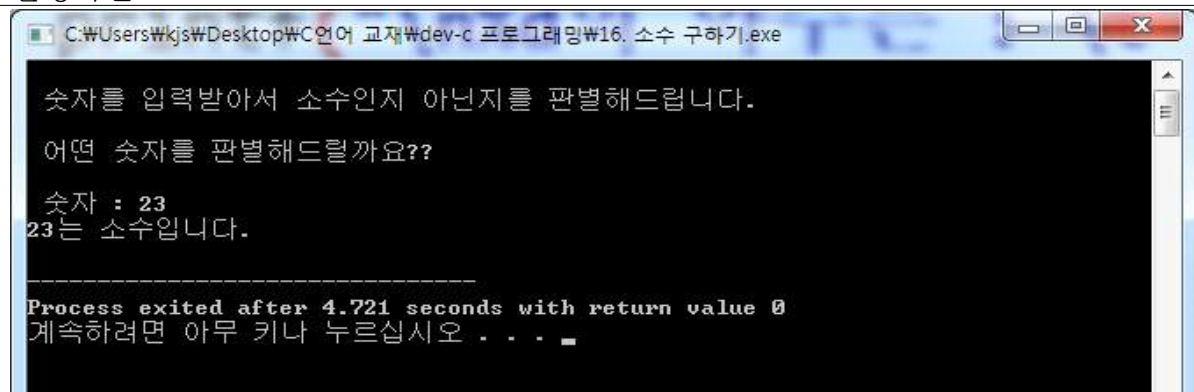
    for(i=1;i<=a;i=i+1)
    {
        if(a%i == 0)
        {
            cnt = cnt + 1;
        }
    }
    if(cnt == 2)
    {
        printf("%d는 소수입니다.\n", a);
        return 0;
    }
    cnt = 0;

    printf("\n%d의 약수는 : \t", a);
```


코드-뒷부분

```
for(i=1;i<=a;i=i+1)
{
    if(a%i == 0)
    {
        printf(" %d\t ", i);
        cnt = cnt +1;
    }
}
printf("입니다");
printf("로.....\n %d가 가진 약수는 모두 %d개입니다." ,a, cnt);
printf("\n따라서 %d는 소수가 아닙니다. \n", a);
}
```

실행화면-1

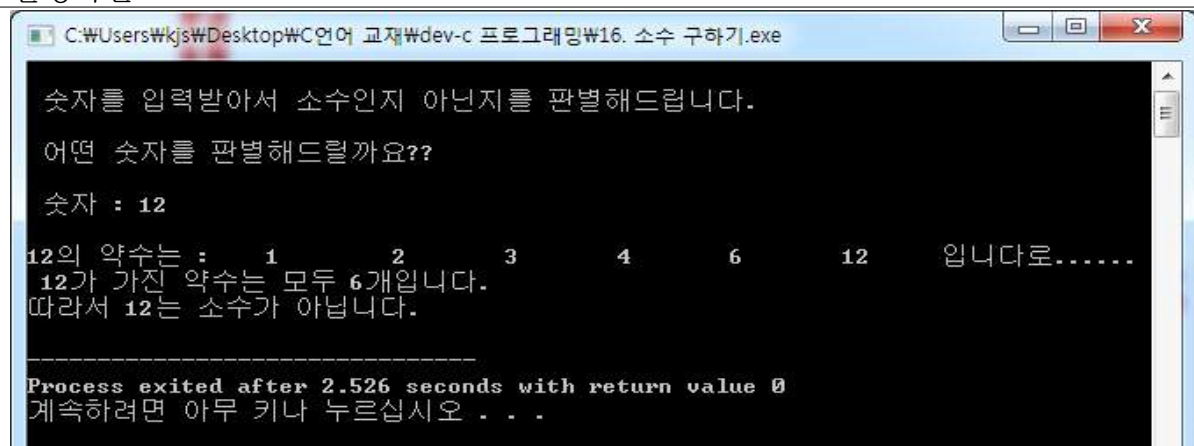


```
C:\Users\Wkjs\Desktop\C언어 교재\dev-c 프로그래밍\16. 소수 구하기.exe

숫자를 입력받아서 소수인지 아닌지를 판별해드립니다.
어떤 숫자를 판별해드릴까요??
숫자 : 23
23는 소수입니다.

-----
Process exited after 4.721 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . . .
```

실행화면-2



```
C:\Users\Wkjs\Desktop\C언어 교재\dev-c 프로그래밍\16. 소수 구하기.exe

숫자를 입력받아서 소수인지 아닌지를 판별해드립니다.
어떤 숫자를 판별해드릴까요??
숫자 : 12
12의 약수는 : 1 2 3 4 6 12 입니다로.....
12가 가진 약수는 모두 6개입니다.
따라서 12는 소수가 아닙니다.

-----
Process exited after 2.526 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . . .
```

17. 반복문 과정 분석하기

코드

```
#include <stdio.h>

int main()
{
    int i = 0;
    int sum = 0;

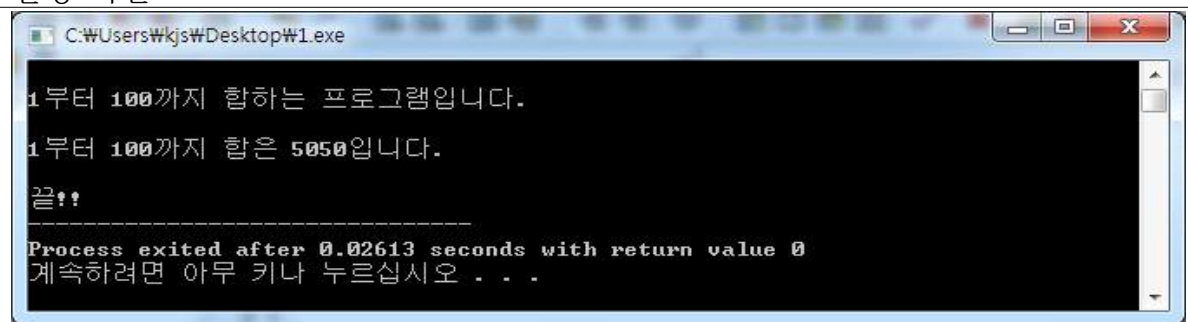
    printf("\n1부터 100까지 합하는 프로그램입니다.\n");

    for(i=1;i<=100;i=i+1)
    {
        sum = sum + i;
    }

    printf("\n1부터 100까지 합은 %d입니다.", sum);
    printf("\n\n끝!!");
}
```

위 코드에서 결국 얻고자하는 결과값은 sum에 저장되어 있습니다.

실행 화면



```
C:\Users\kjs\Desktop\1.exe

1부터 100까지 합하는 프로그램입니다.
1부터 100까지 합은 5050입니다.
끝!!
-----
Process exited after 0.02613 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

하지만 C언어를 입문하면서 생기는 궁금증들이 여러 가지가 있을 것입니다. sum의 누적과정이라든가 i는 어떤 값을 저장하고 for(반복문)이 끝나게 되는 것인지 알아볼 필요가 있습니다. 또한 반복문은 몇 번 반복하게 되는 것인지도 알아보도록 하겠습니다.

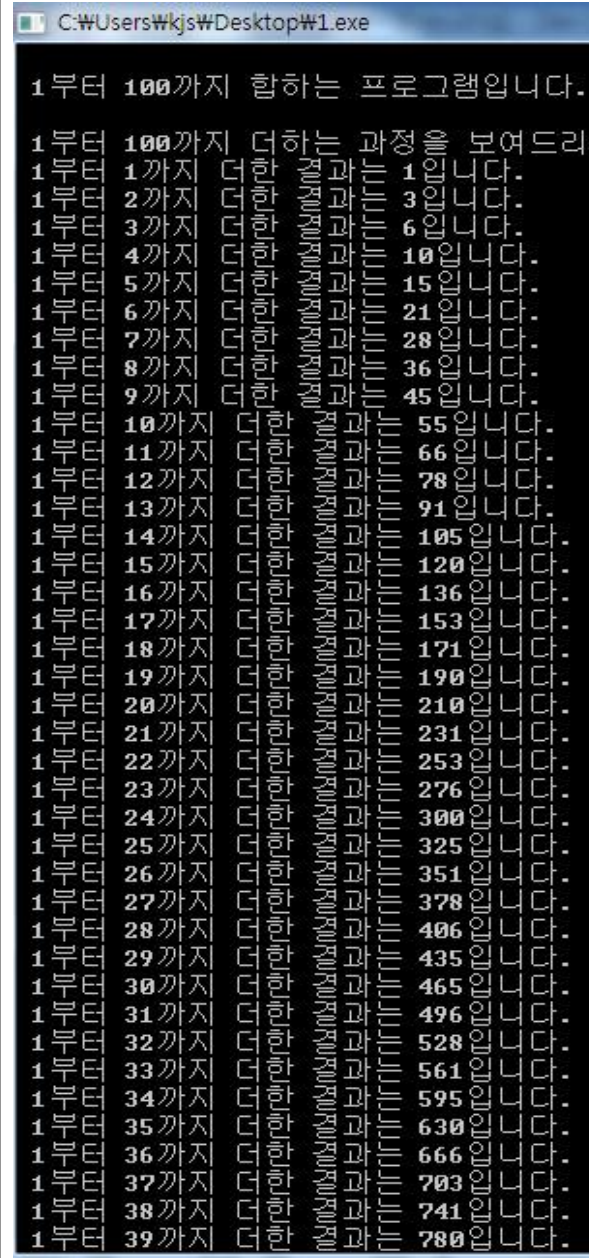
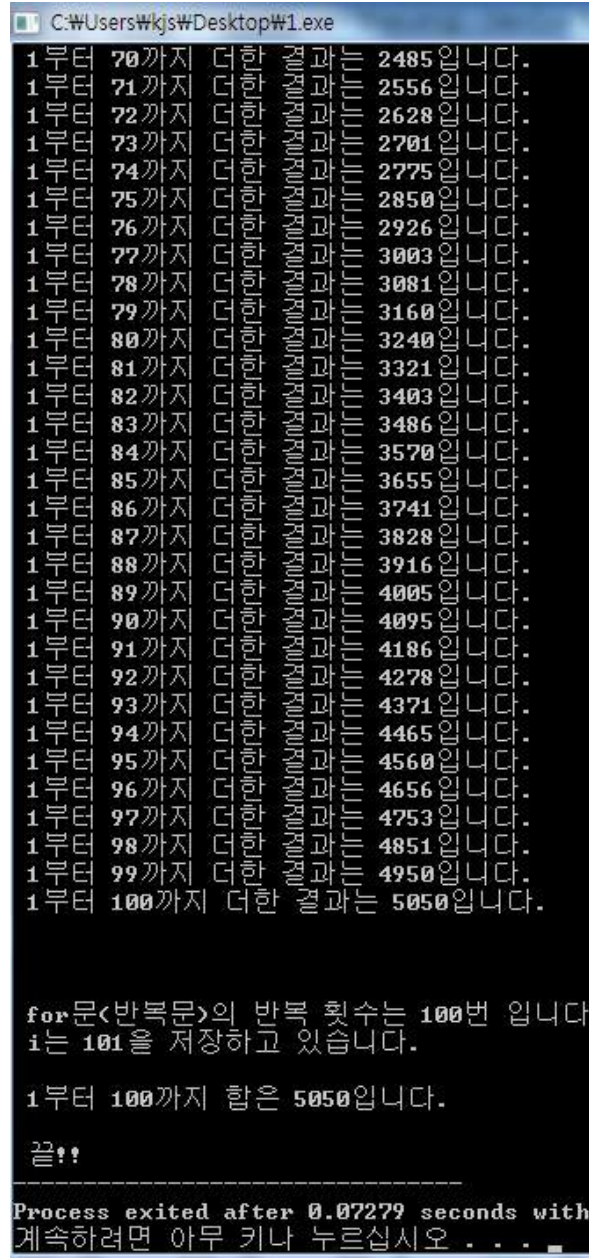
수정 코드

```
#include <stdio.h>

int main()
{
    int i, sum = 0;
    int cnt = 0; //반복문의 반복 횟수 카운팅

    printf("\n 1부터 100까지 합하는 프로그램입니다.\n");
    printf("\n 1부터 100까지 더하는 과정을 보여드리겠습니다.\n");

    for(i=1;i<=100;i=i+1)
    {
        sum = sum + i;
        cnt = cnt + 1;
        printf(" 1부터 %d까지 더한 결과는 %d입니다.\n", i, sum);
    }
    printf("\n\n for문(반복문)의 반복 횟수는 %d번 입니다.", cnt);
    printf("\n i는 %d을 저장하고 있습니다.", i);
    printf("\n\n 1부터 100까지 합은 %d입니다.", sum);
    printf("\n\n 끝!!");
}
```

수정 코드의 실행 화면 - 윗부분	수정 코드의 실행 화면 - 아랫부분
 <pre> 1부터 100까지 합하는 프로그램입니다. 1부터 100까지 더한 과정을 보여드리겠습니다. 1부터 1까지 더한 결과는 1입니다. 1부터 2까지 더한 결과는 3입니다. 1부터 3까지 더한 결과는 6입니다. 1부터 4까지 더한 결과는 10입니다. 1부터 5까지 더한 결과는 15입니다. 1부터 6까지 더한 결과는 21입니다. 1부터 7까지 더한 결과는 28입니다. 1부터 8까지 더한 결과는 36입니다. 1부터 9까지 더한 결과는 45입니다. 1부터 10까지 더한 결과는 55입니다. 1부터 11까지 더한 결과는 66입니다. 1부터 12까지 더한 결과는 78입니다. 1부터 13까지 더한 결과는 91입니다. 1부터 14까지 더한 결과는 105입니다. 1부터 15까지 더한 결과는 120입니다. 1부터 16까지 더한 결과는 136입니다. 1부터 17까지 더한 결과는 153입니다. 1부터 18까지 더한 결과는 171입니다. 1부터 19까지 더한 결과는 190입니다. 1부터 20까지 더한 결과는 210입니다. 1부터 21까지 더한 결과는 231입니다. 1부터 22까지 더한 결과는 253입니다. 1부터 23까지 더한 결과는 276입니다. 1부터 24까지 더한 결과는 300입니다. 1부터 25까지 더한 결과는 325입니다. 1부터 26까지 더한 결과는 351입니다. 1부터 27까지 더한 결과는 378입니다. 1부터 28까지 더한 결과는 406입니다. 1부터 29까지 더한 결과는 435입니다. 1부터 30까지 더한 결과는 465입니다. 1부터 31까지 더한 결과는 496입니다. 1부터 32까지 더한 결과는 528입니다. 1부터 33까지 더한 결과는 561입니다. 1부터 34까지 더한 결과는 595입니다. 1부터 35까지 더한 결과는 630입니다. 1부터 36까지 더한 결과는 666입니다. 1부터 37까지 더한 결과는 703입니다. 1부터 38까지 더한 결과는 741입니다. 1부터 39까지 더한 결과는 780입니다. </pre>	 <pre> 1부터 70까지 더한 결과는 2485입니다. 1부터 71까지 더한 결과는 2556입니다. 1부터 72까지 더한 결과는 2628입니다. 1부터 73까지 더한 결과는 2701입니다. 1부터 74까지 더한 결과는 2775입니다. 1부터 75까지 더한 결과는 2850입니다. 1부터 76까지 더한 결과는 2926입니다. 1부터 77까지 더한 결과는 3003입니다. 1부터 78까지 더한 결과는 3081입니다. 1부터 79까지 더한 결과는 3160입니다. 1부터 80까지 더한 결과는 3240입니다. 1부터 81까지 더한 결과는 3321입니다. 1부터 82까지 더한 결과는 3403입니다. 1부터 83까지 더한 결과는 3486입니다. 1부터 84까지 더한 결과는 3570입니다. 1부터 85까지 더한 결과는 3655입니다. 1부터 86까지 더한 결과는 3741입니다. 1부터 87까지 더한 결과는 3828입니다. 1부터 88까지 더한 결과는 3916입니다. 1부터 89까지 더한 결과는 4005입니다. 1부터 90까지 더한 결과는 4095입니다. 1부터 91까지 더한 결과는 4186입니다. 1부터 92까지 더한 결과는 4278입니다. 1부터 93까지 더한 결과는 4371입니다. 1부터 94까지 더한 결과는 4465입니다. 1부터 95까지 더한 결과는 4560입니다. 1부터 96까지 더한 결과는 4656입니다. 1부터 97까지 더한 결과는 4753입니다. 1부터 98까지 더한 결과는 4851입니다. 1부터 99까지 더한 결과는 4950입니다. 1부터 100까지 더한 결과는 5050입니다. for문<반복문>의 반복 횟수는 100번 입니다. i는 101을 저장하고 있습니다. 1부터 100까지 합은 5050입니다. 끝!! ----- Process exited after 0.07279 seconds with 계속하려면 아무 키나 누르십시오 . . . </pre>

위의 결과를 통해 알아볼 수 있는 의미있는 결과는 다음과 같습니다.

sum의 누적과정 : 누적이라는 개념을 확실히 확인 가능합니다.

i는 100으로 끝나는 것이 아니라 101이 되어서 반복문을 빠져나오게 됩니다. 즉 100을 초과하게 되면 더 이상 반복문(for)을 실행하지 않고 반복문의 다음 코드를 실행하게 되는 것입니다.

그리고 마지막으로 반복문은 우리의 예상과 같이 100번 몇 번 반복하게 되는 것을 확인할 수 있었습니다.

18. 두 수의 차이 구하기

코드

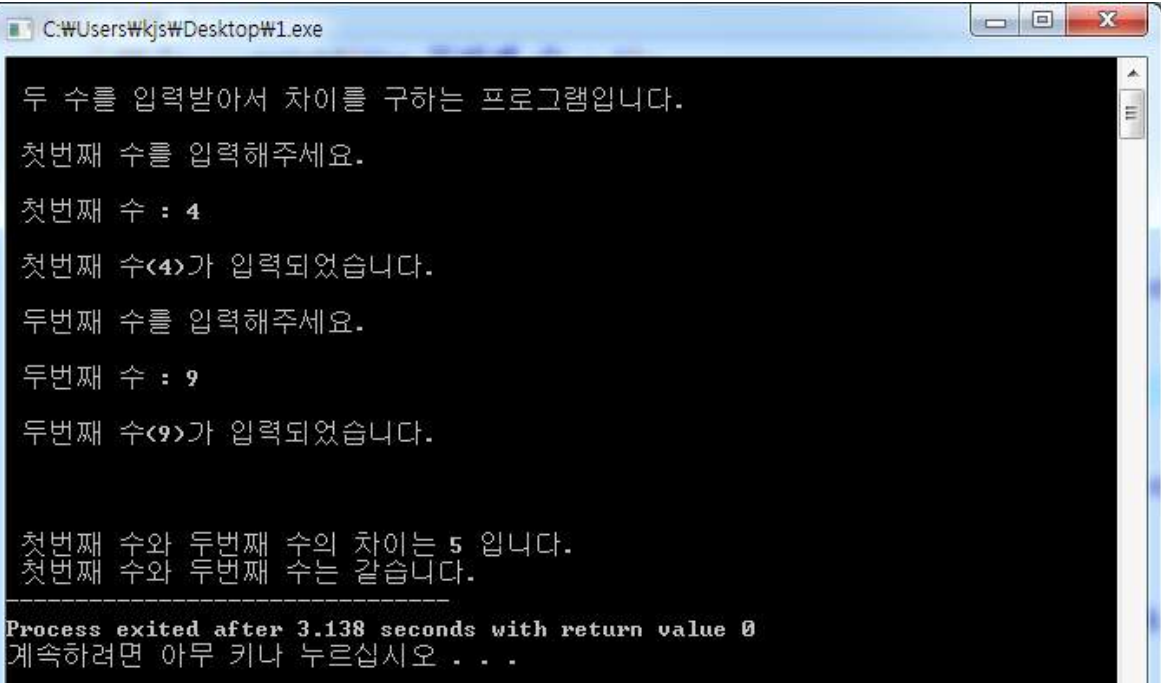
```
#include <stdio.h>
int main()
{
    int a, b = 0;
    int c = 0;

    printf("\n 두 수를 입력받아서 차이를 구하는 프로그램입니다.\n");
    printf("\n 첫번째 수를 입력해주세요.\n");
    printf("\n 첫번째 수 : ");
    scanf("%d", &a);
    printf("\n 첫번째 수(%d)가 입력되었습니다.\n", a);

    printf("\n 두번째 수를 입력해주세요.\n");
    printf("\n 두번째 수 : ");
    scanf("%d", &b);
    printf("\n 두번째 수(%d)가 입력되었습니다.\n", b);

    if(a > b)
    {
        c = a - b;
        printf("\n\n\n 첫번째 수와 두번째 수의 차이는 %d 입니다.", c);
    }
    else if(a < b)
    {
        c = b - a;
        printf("\n\n\n 첫번째 수와 두번째 수의 차이는 %d 입니다.", c);
    }
    else
    {
        printf("\n\n\n 첫번째 수와 두번째 수의 차이는 없습니다.");
        printf("\n 첫번째 수와 두번째 수는 같습니다.");
    }
}
```

실행 화면 - 입력 : 4, 9

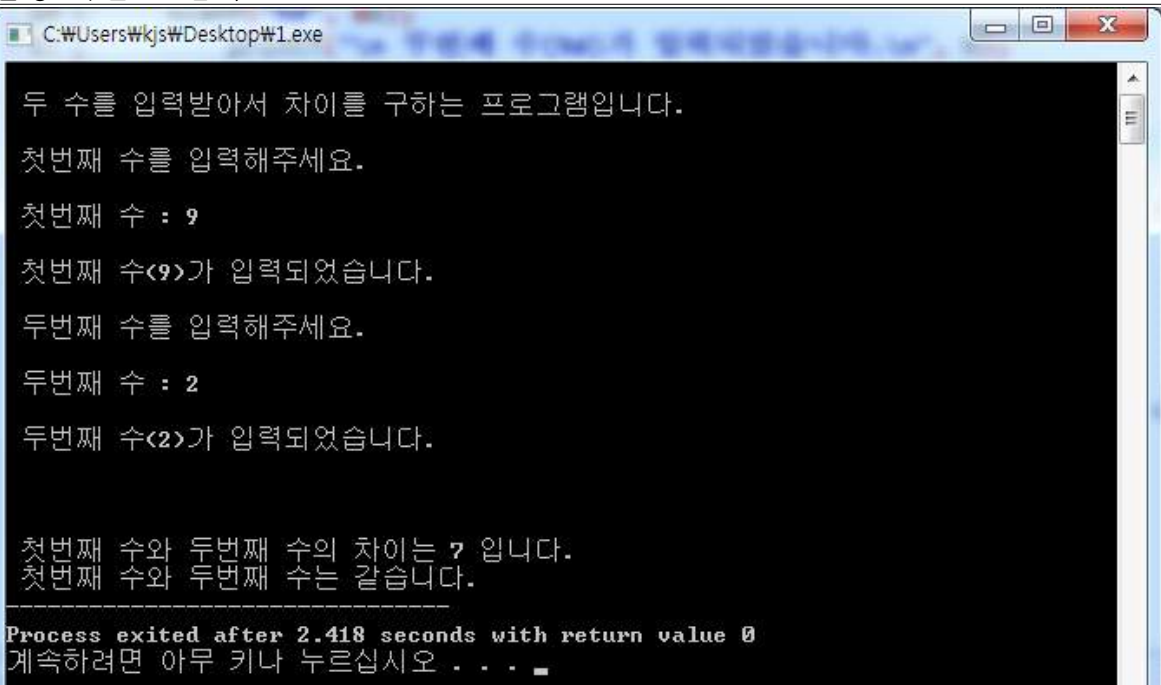


```
C:\Users\Wkjs\Desktop\W1.exe

두 수를 입력받아서 차이를 구하는 프로그램입니다.
첫번째 수를 입력해주세요.
첫번째 수 : 4
첫번째 수<4>가 입력되었습니다.
두번째 수를 입력해주세요.
두번째 수 : 9
두번째 수<9>가 입력되었습니다.

첫번째 수와 두번째 수의 차이는 5 입니다.
첫번째 수와 두번째 수는 같습니다.
-----
Process exited after 3.138 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

실행 화면 - 입력 : 9, 2



```
C:\Users\Wkjs\Desktop\W1.exe

두 수를 입력받아서 차이를 구하는 프로그램입니다.
첫번째 수를 입력해주세요.
첫번째 수 : 9
첫번째 수<9>가 입력되었습니다.
두번째 수를 입력해주세요.
두번째 수 : 2
두번째 수<2>가 입력되었습니다.

첫번째 수와 두번째 수의 차이는 7 입니다.
첫번째 수와 두번째 수는 같습니다.
-----
Process exited after 2.418 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

두 개의 수를 입력받아서 두 수의 차이를 표현해주는 기능을 잘 하는 코드입니다. 하지만 첫 번째 수와 두 번째 수가 같다는 printf()문장은 왜 계속 나오는 것일까요??

이 문제를 해결하기 위해 아래와 같이 코딩을 해보겠습니다.

수정 코드

```
#include <stdio.h>

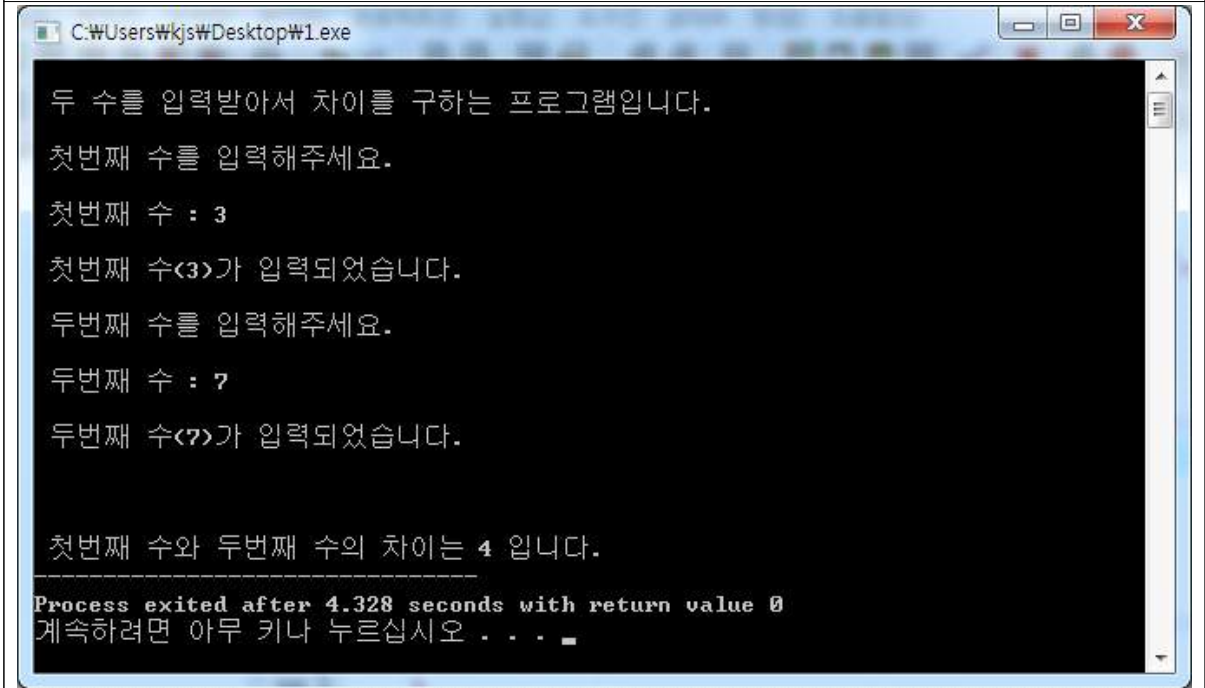
int main()
{
    int a, b = 0;
    int c = 0;

    printf("\n 두 수를 입력받아서 차이를 구하는 프로그램입니다.\n");
    printf("\n 첫번째 수를 입력해주세요.\n");
    printf("\n 첫번째 수 : ");
    scanf("%d", &a);
    printf("\n 첫번째 수(%d)가 입력되었습니다.\n", a);

    printf("\n 두번째 수를 입력해주세요.\n");
    printf("\n 두번째 수 : ");
    scanf("%d", &b);
    printf("\n 두번째 수(%d)가 입력되었습니다.\n", b);

    if(a > b)
    {
        c = a - b;
        printf("\n\n\n 첫번째 수와 두번째 수의 차이는 %d 입니다.", c);
        return 0;
    }
    else if(a < b)
    {
        c = b - a;
        printf("\n\n\n 첫번째 수와 두번째 수의 차이는 %d 입니다.", c);
        return 0;
    }
    else
    {
        printf("\n\n\n 첫번째 수와 두번째 수의 차이는 없습니다.");
        printf("\n 첫번째 수와 두번째 수는 같습니다.");
    }
}
```

수정 코드 실행 화면 - 입력 : 3, 7



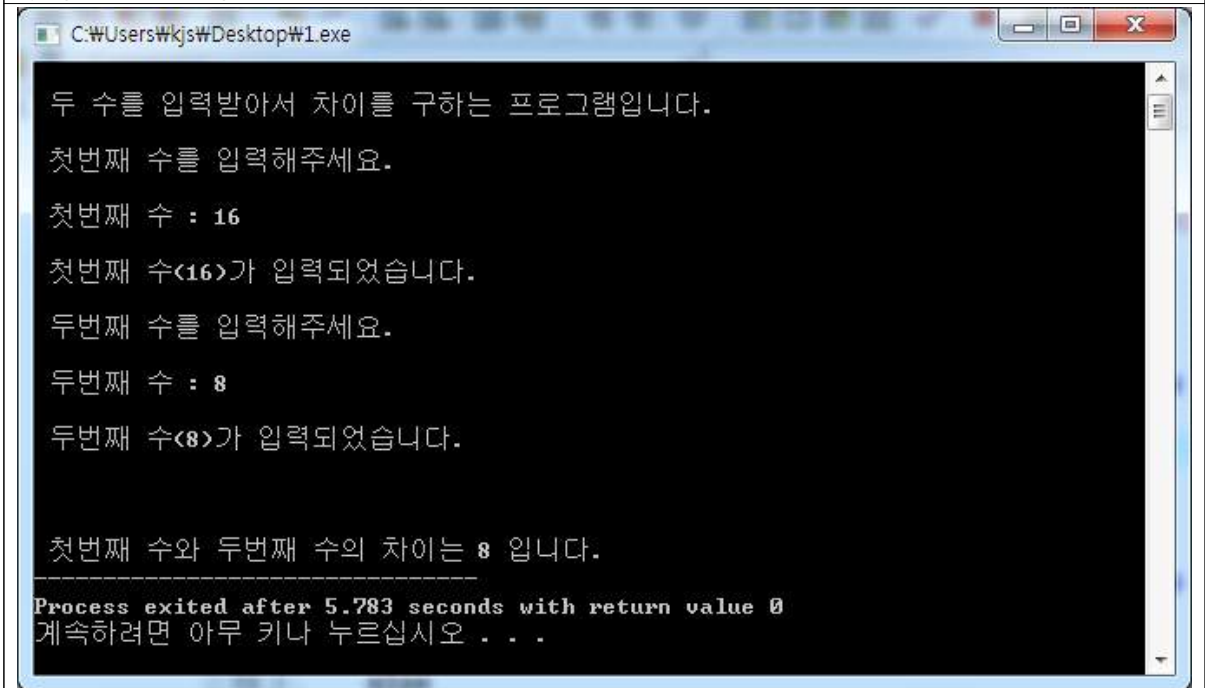
```
C:\Users\Wjs\Desktop\W1.exe

두 수를 입력받아서 차이를 구하는 프로그램입니다.
첫번째 수를 입력해주세요.
첫번째 수 : 3
첫번째 수<3>가 입력되었습니다.
두번째 수를 입력해주세요.
두번째 수 : 7
두번째 수<7>가 입력되었습니다.

첫번째 수와 두번째 수의 차이는 4 입니다.

-----
Process exited after 4.328 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

수정 코드 실행 화면 - 입력 : 16, 8



```
C:\Users\Wjs\Desktop\W1.exe

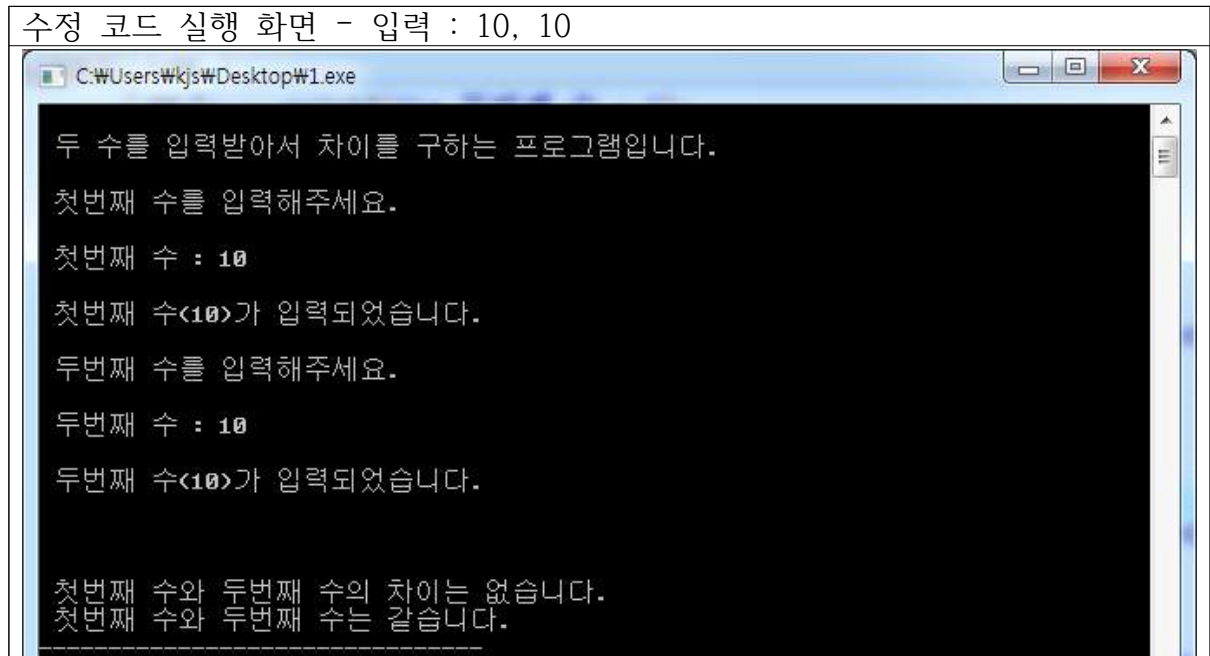
두 수를 입력받아서 차이를 구하는 프로그램입니다.
첫번째 수를 입력해주세요.
첫번째 수 : 16
첫번째 수<16>가 입력되었습니다.
두번째 수를 입력해주세요.
두번째 수 : 8
두번째 수<8>가 입력되었습니다.

첫번째 수와 두번째 수의 차이는 8 입니다.

-----
Process exited after 5.783 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```


그럼 마지막으로 같은 수를 입력해보겠습니다.

수정 코드 실행 화면 - 입력 : 10, 10

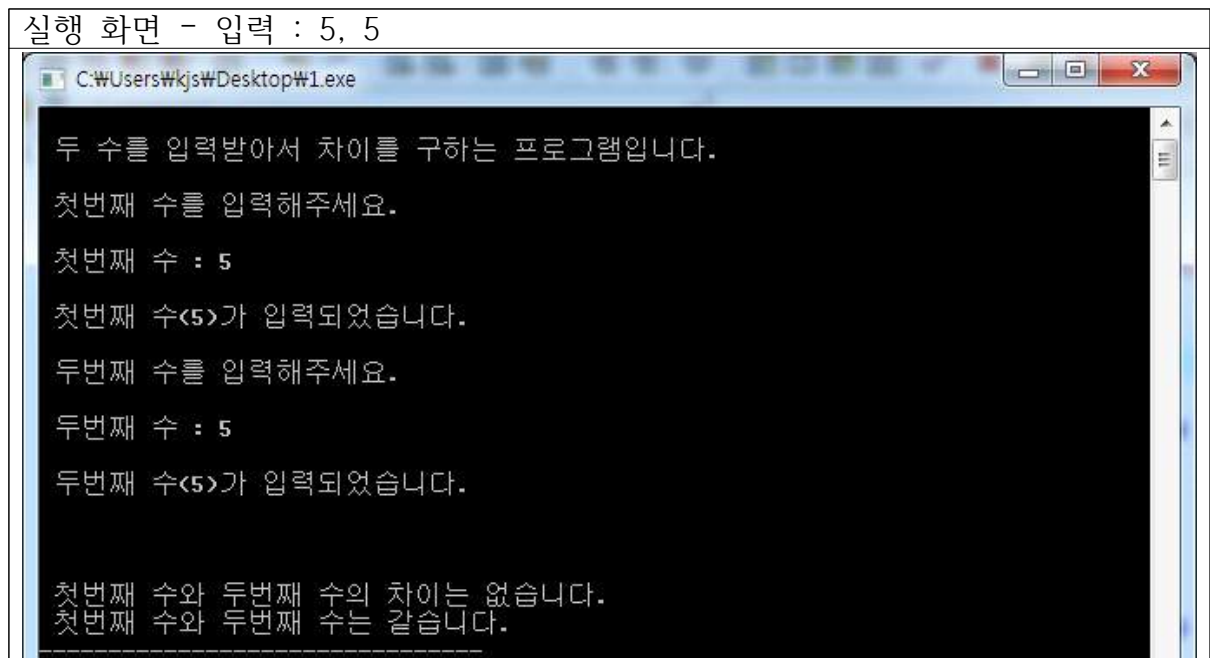


```
C:\Users\Wkjs\Desktop\W1.exe
두 수를 입력받아서 차이를 구하는 프로그램입니다.
첫번째 수를 입력해주세요.
첫번째 수 : 10
첫번째 수<10>가 입력되었습니다.
두번째 수를 입력해주세요.
두번째 수 : 10
두번째 수<10>가 입력되었습니다.

첫번째 수와 두번째 수의 차이는 없습니다.
첫번째 수와 두번째 수는 같습니다.
```

수정 전 코드에서 같은 수를 넣으면 어떨까요?

실행 화면 - 입력 : 5, 5



```
C:\Users\Wkjs\Desktop\W1.exe
두 수를 입력받아서 차이를 구하는 프로그램입니다.
첫번째 수를 입력해주세요.
첫번째 수 : 5
첫번째 수<5>가 입력되었습니다.
두번째 수를 입력해주세요.
두번째 수 : 5
두번째 수<5>가 입력되었습니다.

첫번째 수와 두번째 수의 차이는 없습니다.
첫번째 수와 두번째 수는 같습니다.
```

결국 수정 전 코드와 수정 후 코드는 첫 번째 수와 두 번째 수가 같을 때에는 오류를 확인 할 수 없다는 사실을 알게 되었습니다.

따라서 여러 가지 경우에서 오류가 생기기 때문에 코딩을 할 때에는 여러 측면에서 오류를 탐색해보는 노력이 필요합니다.

19. 최소값 찾기 - 초기값 설정 문제

코드

```
#include <stdio.h>
int main()
{
    int a, b, i = 0;
    int min = 100;

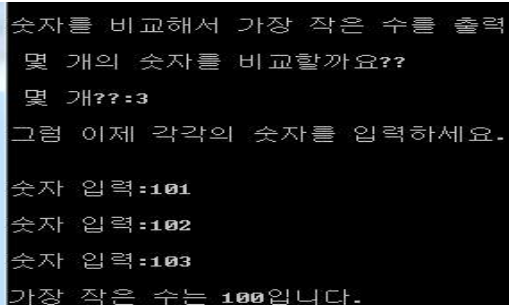
    printf("\n숫자를 비교해서 가장 작은 수를 출력");
    printf("\n\n 몇 개의 숫자를 비교할까요??");
    printf("\n\n 몇 개??");
    scanf("%d",&a);
    printf("\n그럼 이제 각각의 숫자를 입력하세요.\n\n");

    for(i=1;i<=a;i=i+1)
    {
        printf("\n숫자 입력:");
        scanf("\n%d",&b);
        if(min>b)
            min = b;
    }
    printf("\n가장 작은 수는 %d입니다.",min);
}
```

위의 코딩에는 문제가 있습니다. 찾아보셨나요??

네. 맞습니다. 실행까지 문법적인 오류는 없지만 실행 시 입력되는 각각의 숫자가 모두 100을 초과하게 되면 최소값은 그대로 100이 됩니다.

실행 화면 - 입력 : 101, 102, 103



```
숫자를 비교해서 가장 작은 수를 출력
몇 개의 숫자를 비교할까요??
몇 개?? : 3
그럼 이제 각각의 숫자를 입력하세요.
숫자 입력: 101
숫자 입력: 102
숫자 입력: 103
가장 작은 수는 100입니다.
```

그렇다고 사용자가 입력할 값을 예측해서 최소값의 초기값을 아주 큰 수로 설정하는 것도 한계가 있습니다.

그럼 어떻게 해야 할까요??

네. 맞습니다. 처음으로 들어오는 수를 최소값을 저장해두면 문제가 해결됩니다.

위와 같이 반복문 안쪽에서 먼저 한 번 입력을 받아서 최소값을 첫 입력값으로 저장하시면 됩니다.

그럼, 문제가 해결되었는지 확인해보겠습니다.

```
실행 화면 - 입력 : 1000, 10000, 100000
C:\Users\Wkjs\Desktop\이름없음1.exe
숫자를 비교해서 가장 작은 수를 출력
몇 개의 숫자를 비교할까요??
몇 개?:3
그럼 이제 각각의 숫자를 입력하세요.

숫자 입력:1000
숫자 입력:10000
숫자 입력:100000
가장 작은 수는 1000입니다.
-----
Process exited after 7.96 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

확인하시는 바와 같이 문제가 해결이 되었습니다.

이렇게 코딩을 하시면 min의 초기값을 설정할 필요가 없습니다. 다만 주의할 점은 반복문의 반복 횟수를 1회 줄여줘야 합니다.

그래서 for(i=1; i<=a-1; i=i+1)와 같이 i<=a;을 i<=a-1;로 변경해야 합니다.

이런 코딩은 최소값 코딩뿐만 아니라 최대값을 코딩할 때에도 사용하게 됩니다.

20. 완전수 구하기

```
코드
#include <stdio.h>

int main()
{
    int num = 0;
    int sum = 0;
    int i;

    printf("완전수를 구하는 프로그램입니다.\n\n");
    printf("완전수는 자기 자신을 제외한 모든 약수의 합이\n ");
    printf("자기 자신과 같은 수를 뜻합니다.\n\n");
    printf("예시 : 6의 약수는 1,2,3,6으로 6을 제외한 1 + 2 + 3이\n");
    printf("6과 같으므로 완전수라고 할 수 있습니다.\n");
    printf("6, 28, 496, 8128은 완전수입니다.\n");

    printf("완전수인지 판별하고자 하는 수를 입력하세요.\n\n\n");
    printf("수 입력 : ");
    scanf("\n %d", &num);

    for(i=1; i<num; i++)    //자기 자신은 제외해야 합니다.
    {
        if( (num%i) == 0)
        {
            sum = sum + i;
        }
    }
    if(sum == num)
    {
        printf("\n\n%d은 완전수입니다.\n", num);
    }
    else
    {
        printf("\n\n%d은 완전수가 아닙니다.\n", num);
    }
}
```

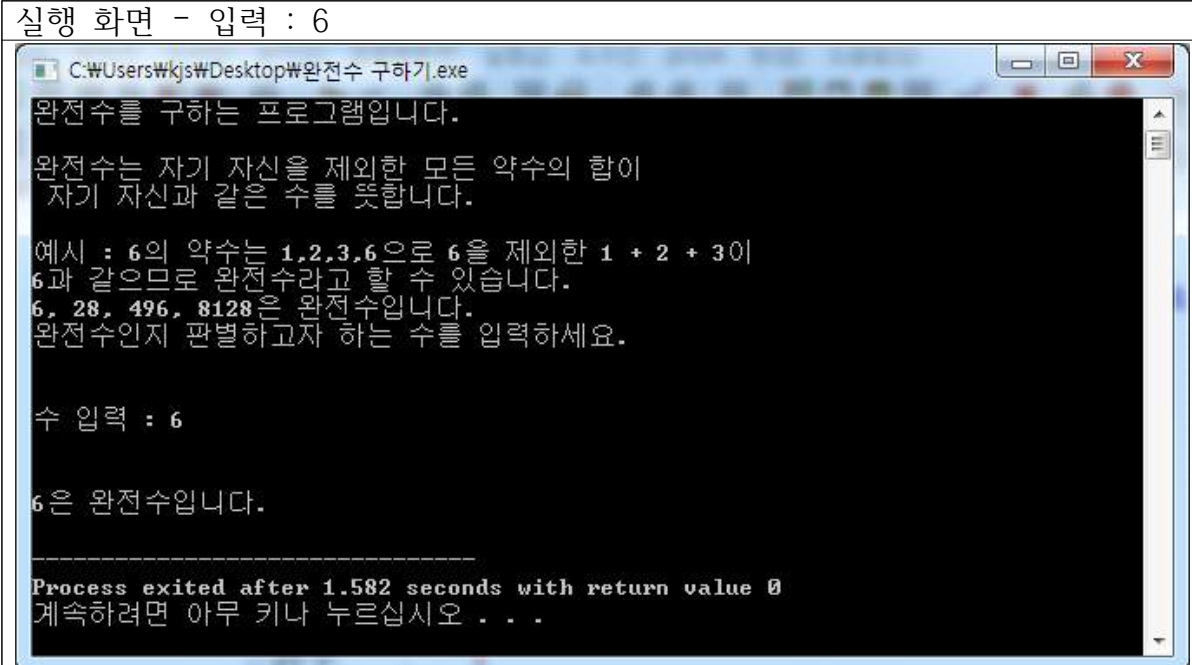
위 코드를 보면 코드의 첫 부분에서 완전수가 무엇인지 자세히 설명하고 있습니다. 이와 같이 프로그래밍을 할 때에는 이 프로그램을 읽는 사람에 대한 배려가 필요함

니다. 완전수를 전혀 알지 못하는 사람도 이 프로그램을 사용할 수 있어야하기 때문입니다.

예시 속에는 6과 28, 496과 8128은 완전수라고 제시되어 있습니다.

그래서 일단 6과 28을 입력해보겠습니다.

실행 화면 - 입력 : 6



```
C:\Users\wkjs\Desktop\완전수 구하기.exe
완전수를 구하는 프로그램입니다.

완전수는 자기 자신을 제외한 모든 약수의 합이
자기 자신과 같은 수를 뜻합니다.

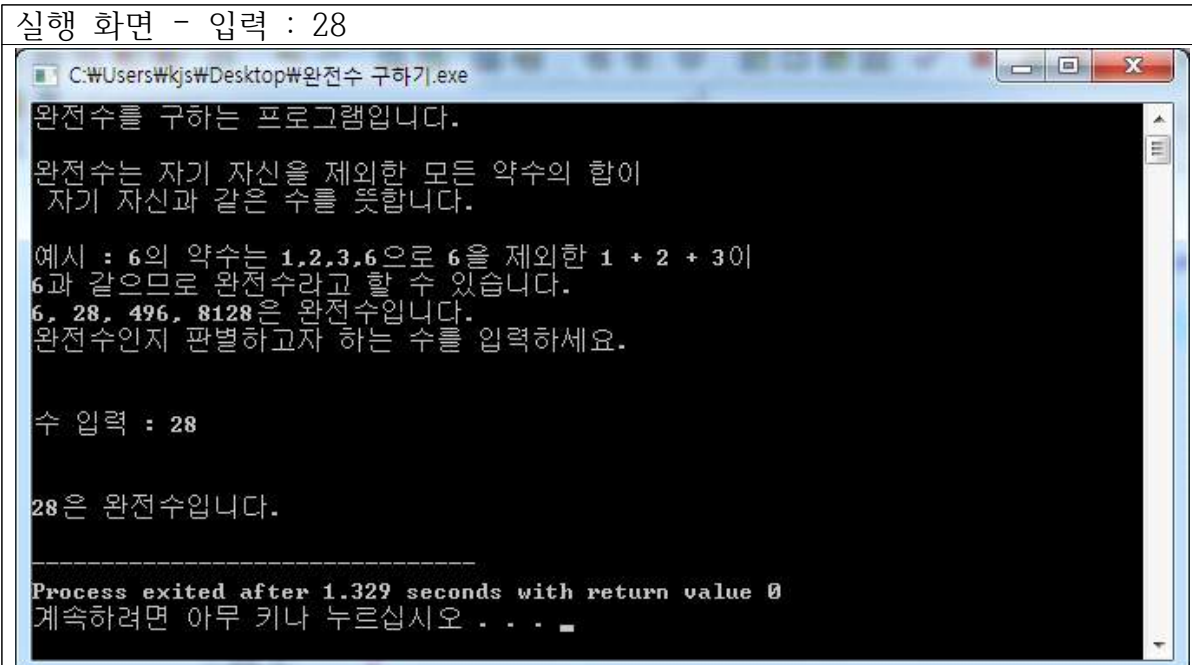
예시 : 6의 약수는 1,2,3,6으로 6을 제외한 1 + 2 + 3이
6과 같으므로 완전수라고 할 수 있습니다.
6, 28, 496, 8128은 완전수입니다.
완전수인지 판별하고자 하는 수를 입력하세요.

수 입력 : 6

6은 완전수입니다.

-----
Process exited after 1.582 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

실행 화면 - 입력 : 28



```
C:\Users\wkjs\Desktop\완전수 구하기.exe
완전수를 구하는 프로그램입니다.

완전수는 자기 자신을 제외한 모든 약수의 합이
자기 자신과 같은 수를 뜻합니다.

예시 : 6의 약수는 1,2,3,6으로 6을 제외한 1 + 2 + 3이
6과 같으므로 완전수라고 할 수 있습니다.
6, 28, 496, 8128은 완전수입니다.
완전수인지 판별하고자 하는 수를 입력하세요.

수 입력 : 28

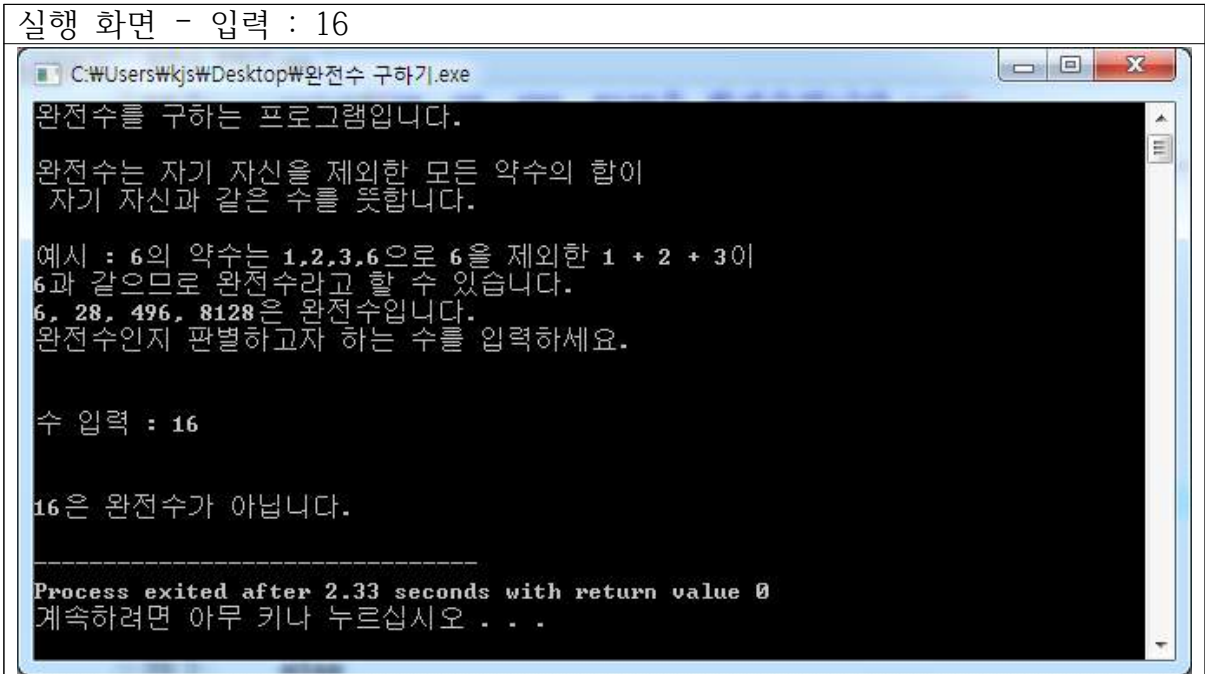
28은 완전수입니다.

-----
Process exited after 1.329 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

혹시 모든 수가 완전수로 출력되는 것은 아닐까요?? 이런 의심이 멋진 프로그램을 만들고 보이지 않는 오류를 찾아내는 원동력이 됩니다.

그러면 확인을 위해 16과 100을 입력해보겠습니다.

실행 화면 - 입력 : 16



```
C:\Users\Wkjs\Desktop\완전수 구하기.exe
완전수를 구하는 프로그램입니다.

완전수는 자기 자신을 제외한 모든 약수의 합이
자기 자신과 같은 수를 뜻합니다.

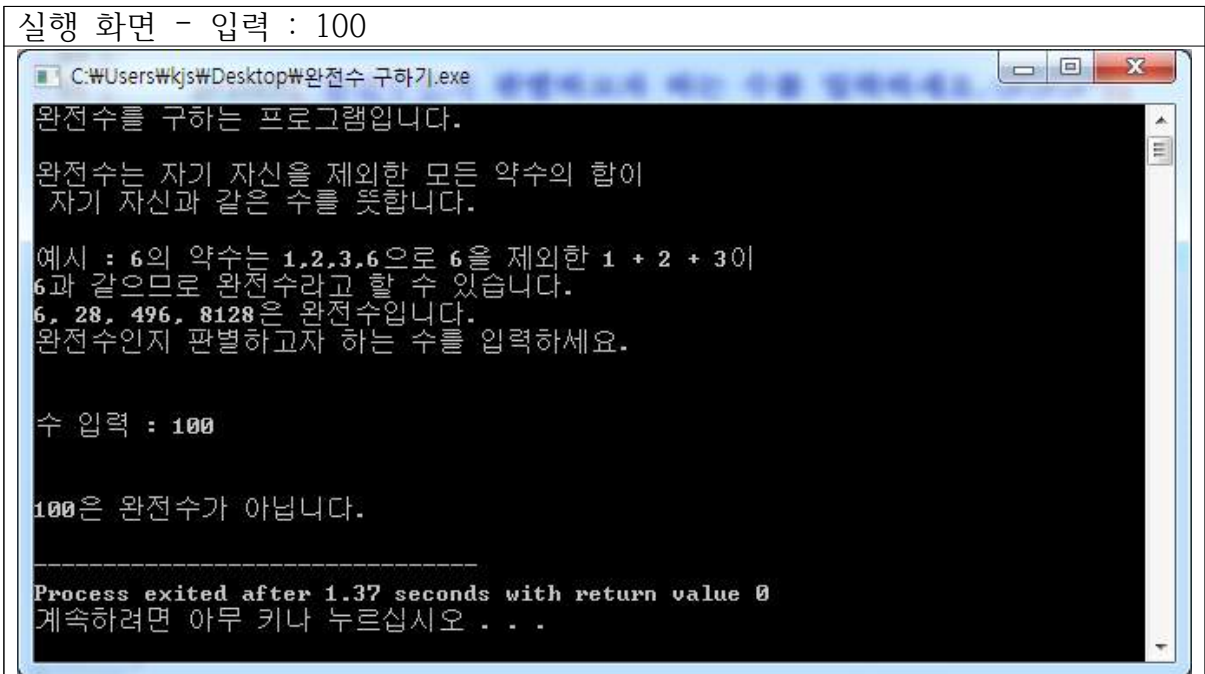
예시 : 6의 약수는 1,2,3,6으로 6을 제외한 1 + 2 + 3이
6과 같으므로 완전수라고 할 수 있습니다.
6, 28, 496, 8128은 완전수입니다.
완전수인지 판별하고자 하는 수를 입력하세요.

수 입력 : 16

16은 완전수가 아닙니다.

-----
Process exited after 2.33 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

실행 화면 - 입력 : 100



```
C:\Users\Wkjs\Desktop\완전수 구하기.exe
완전수를 구하는 프로그램입니다.

완전수는 자기 자신을 제외한 모든 약수의 합이
자기 자신과 같은 수를 뜻합니다.

예시 : 6의 약수는 1,2,3,6으로 6을 제외한 1 + 2 + 3이
6과 같으므로 완전수라고 할 수 있습니다.
6, 28, 496, 8128은 완전수입니다.
완전수인지 판별하고자 하는 수를 입력하세요.

수 입력 : 100

100은 완전수가 아닙니다.

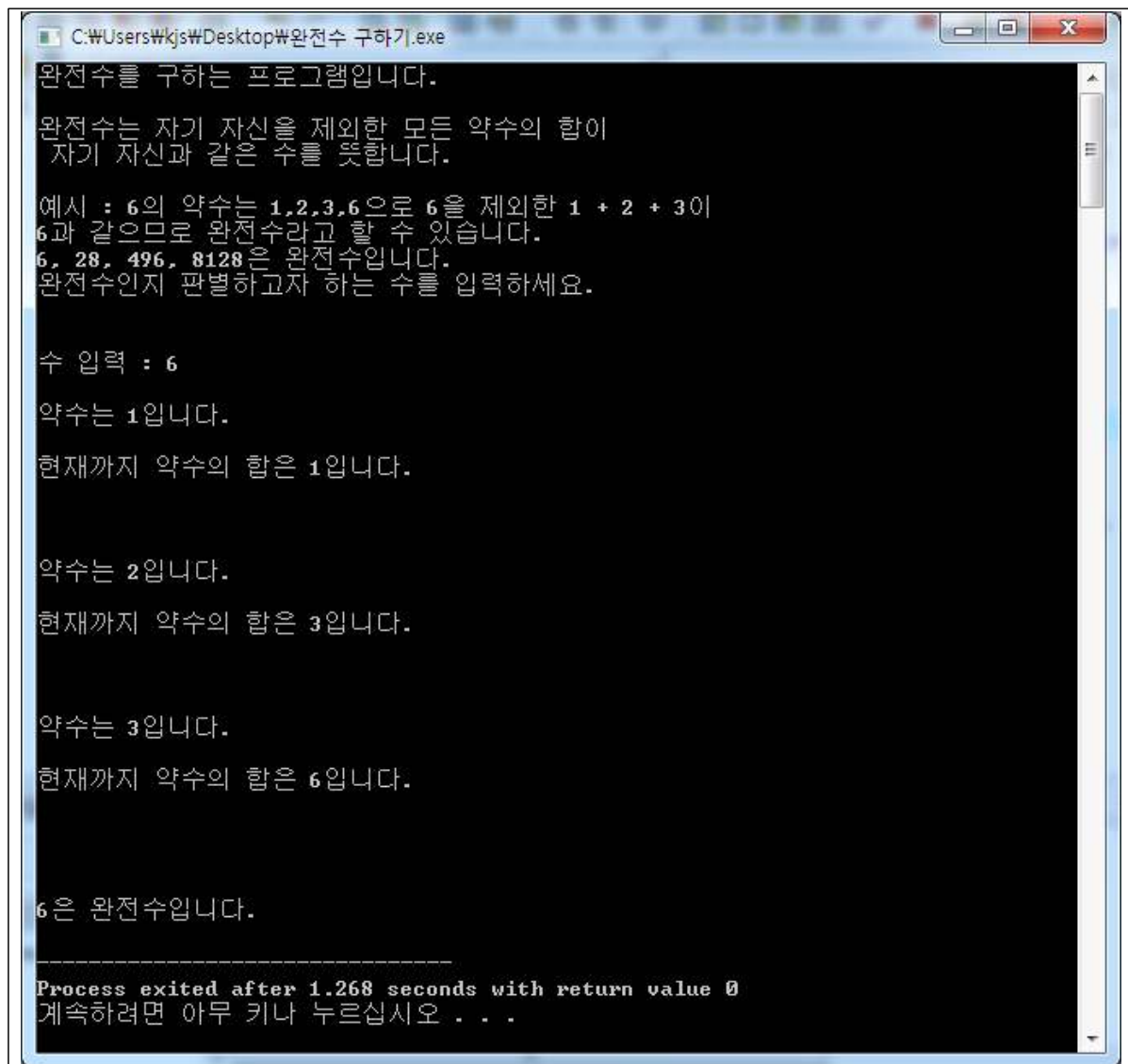
-----
Process exited after 1.37 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

어떤 원리로 완전수가 되는 것인지, 어떤 과정으로 완전수가 되지 못한 것인지 궁금하기만 합니다. 그래서 코딩을 조금 바꿔보겠습니다.

for()반복문에 아래와 같이 코드 두 줄을 추가해보겠습니다.

```
for(i=1; i<num; i++)
{
    if((num%i)==0)
    {
        sum = sum + i;
        printf("\n약수는 %d입니다.\n", i);
        printf("\n현재까지 약수의 합은 %d입니다.\n\n\n", sum);
    }
}
```

이렇게 하면 결과는 아래와 같이 나옵니다. (입력 : 6)



```
C:\Users\wkjs\Desktop\완전수 구하기.exe
완전수를 구하는 프로그램입니다.
완전수는 자기 자신을 제외한 모든 약수의 합이
자기 자신과 같은 수를 뜻합니다.

예시 : 6의 약수는 1,2,3,6으로 6을 제외한 1 + 2 + 3이
6과 같으므로 완전수라고 할 수 있습니다.
6, 28, 496, 8128은 완전수입니다.
완전수인지 판별하고자 하는 수를 입력하세요.

수 입력 : 6
약수는 1입니다.
현재까지 약수의 합은 1입니다.

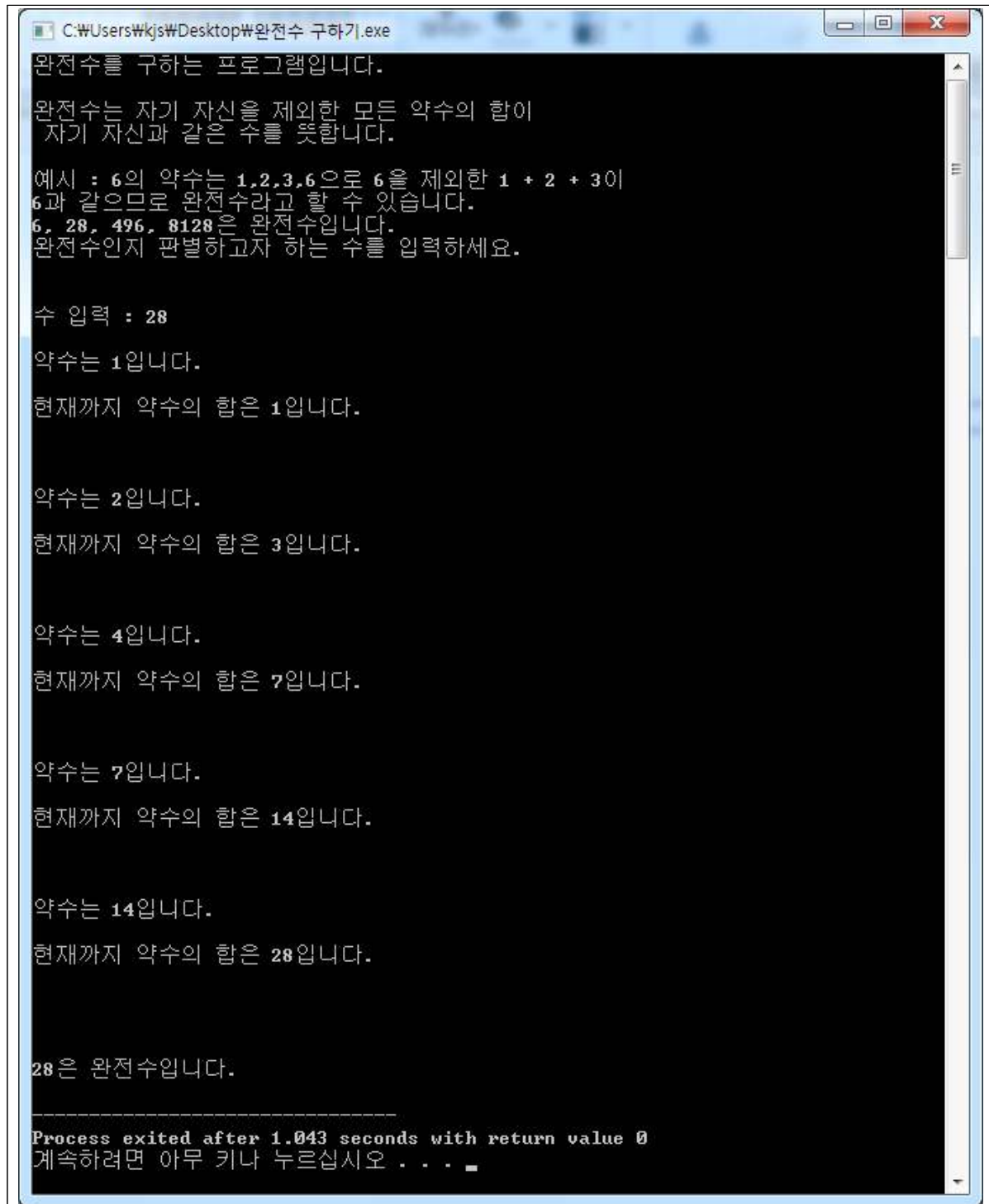
약수는 2입니다.
현재까지 약수의 합은 3입니다.

약수는 3입니다.
현재까지 약수의 합은 6입니다.

6은 완전수입니다.

-----
Process exited after 1.268 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

한 개 더 해보겠습니다. (입력 : 28)



```
C:\Users\Wkjs\Desktop\완전수 구하기.exe
완전수를 구하는 프로그램입니다.

완전수는 자기 자신을 제외한 모든 약수의 합이
자기 자신과 같은 수를 뜻합니다.

예시 : 6의 약수는 1,2,3,6으로 6을 제외한 1 + 2 + 3이
6과 같으므로 완전수라고 할 수 있습니다.
6, 28, 496, 8128은 완전수입니다.
완전수인지 판별하고자 하는 수를 입력하세요.

수 입력 : 28
약수는 1입니다.
현재까지 약수의 합은 1입니다.

약수는 2입니다.
현재까지 약수의 합은 3입니다.

약수는 4입니다.
현재까지 약수의 합은 7입니다.

약수는 7입니다.
현재까지 약수의 합은 14입니다.

약수는 14입니다.
현재까지 약수의 합은 28입니다.

28은 완전수입니다.

-----
Process exited after 1.043 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

역시 반복문의 과정을 자세히 보기 위해서는 반복문 안쪽에 printf()문을 삽입하여 한 단계씩 분석해보는 것이 좋습니다. 각 단계의 과정을 파악할 수 있을 뿐 아니라 오류의 위치도 쉽게 찾아 해결할 수 있기 때문입니다.

21. 1차원 배열 활용하기

배열은 같은 자료형을 연속적으로 저장할 수 있는 데이터형으로 사용하기 편리하고 간단한 장점을 가지고 있다.

다음의 코드를 실행해보자.

코드

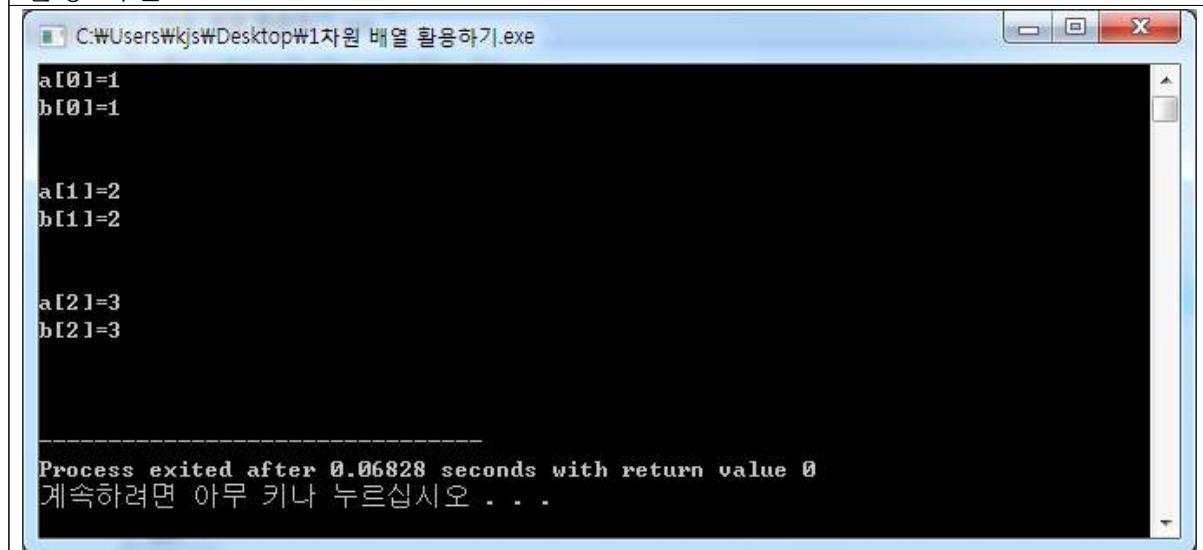
```
#include <stdio.h>

int main()
{
    int a[] = {1, 2, 3};
    int b[3] = {1, 2, 3};
    int i;

    for(i=0; i<3; i++)
    {
        printf("a[%d]=%d\n",i,a[i]);
        printf("b[%d]=%d\n\n",i,b[i]);
    }
}
```

실행 결과도 한 번 볼까요?

실행 화면



```
C:\Users\Wkjs\Desktop\1차원 배열 활용하기.exe
a[0]=1
b[0]=1

a[1]=2
b[1]=2

a[2]=3
b[2]=3

-----
Process exited after 0.06828 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```

배열에도 자료형이 있습니다. 위의 코드는 int로 배열을 정의하고 있습니다. 그렇게

되면 배열에 저장되는 모든 자료는 정수형이 되어야 한다는 뜻입니다.

b[3]처럼 배열의 크기를 미리 정해도 되고 a[]처럼 배열의 크기를 미리 정하지 않아도 됩니다. 다만 미리 정해놓은 크기대로 실행되는 프로그램이 컴퓨터를 덜 힘들게 합니다. 즉 고정적인 정의방식이 가변적인 정의 방식보다 컴퓨터에게 이롭다는 뜻입니다. 하지만 어떻게 보면 인간이 사용하는 방식으로는 가변적인 방식이 더 간편할 수 있다는 생각이 듭니다.

그래도 자신이 사용할 배열의 크기를 미리 정해놓고 사용하는 습관을 길러서 컴퓨터를 더 효율적으로 사용하는 연습을 해야 할 것입니다.

그럼 다른 자료형은 어떻게 사용할까요?

코드

```
#include <stdio.h>

int main()
{
    char a[]={0,};

    printf("\n이름을 입력하시오.\n");
    printf("이름 : ");
    scanf("\n %s",a);

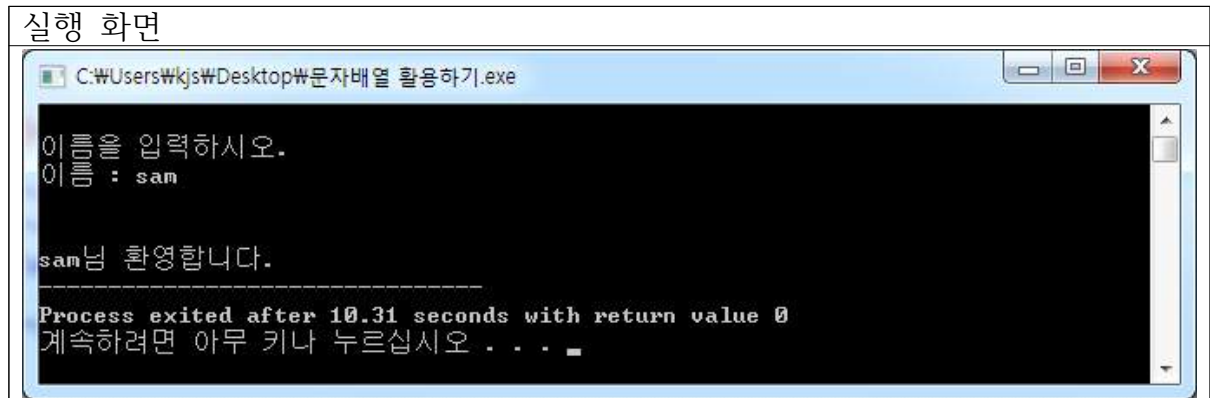
    printf("\n\n%s님 환영합니다.", a);
}
```

char a[]={0,}; 이 문장은 문자를 연속으로 저장하는 a라는 배열을 초기화하는 문장입니다. 몇 개의 문자를 저장할지 모르는 상황에서 쓰레기값이 들어있는 저장공간을 모두 출력할 수는 없으니까요.

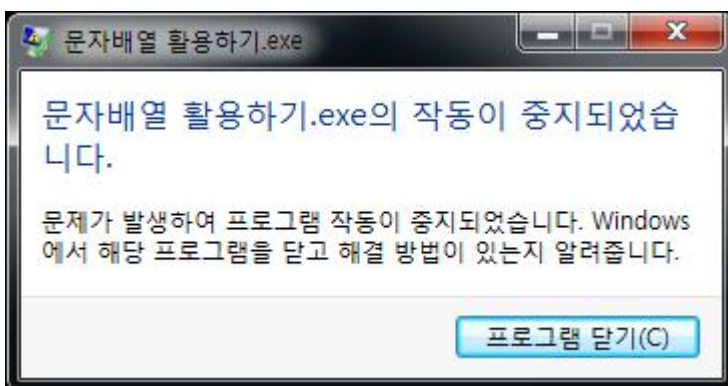
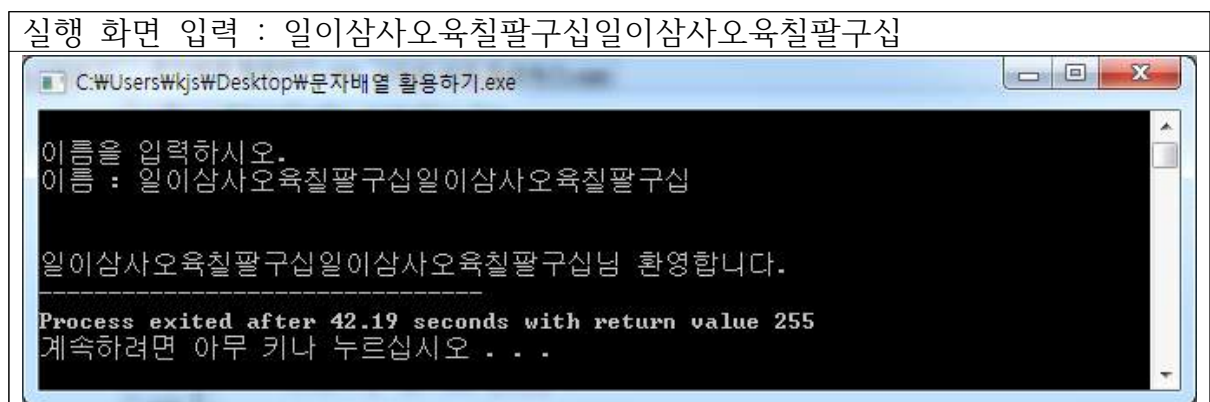
그리고 **scanf("\n %s",a);** 문장에서 주목할 점은 배열의 이름 자체가 포인터라는 점입니다. 본래 &기호를 사용해야 하지만 배열에서는 &기호를 사용하지 않습니다. 마지막으로 %s라는 변환문자를 사용했습니다. 문자는 char로 선언하지만 문자가 연속으로 배열이 되면 문장이 되는 셈입니다. 따라서 출력을 할 때에는 문장형데이터를 표현하는 변환문자인 %s를 사용하게 되는 것입니다.

다시 한 번 자세히 코드를 살펴보고 활용하시면 좋겠습니다.

그럼 위의 코드를 실행해볼까요?



코드에서 `char a[]={0,};` 부분을 `char a[10]={0,};`로 수정해서 실행해보겠습니다. 물론 sam이라는 세 문자를 입력한다면 결과는 위와 같습니다. 하지만 일이삼사오육칠팔구십일이삼사오육칠팔구십처럼 많은 글자를 입력한다면 어떻게 될까요?



위와 같은 실행 결과와 함께 시스템 오류 창이 한꺼번에 발생하는 것을 보니 아무래도 프로그램을 실행시키는 것이 버겁나 보입니다. 따라서 우리는 적당한 공간을 정하고 정해진 공간을 효율적으로 사용하는 프로그래밍을 하는 노력이 필요합니다.

22. 2차원 배열 활용하기

1차원 배열의 논리적 모양은 아래와 같습니다.

a[0]	a[1]	a[2]	a[3]	a[4]

단, 주의해야할 점은 배열의 방 번호가 0번부터 시작한다는 점입니다. 조금 이상할 수도 있겠지만 이론상 2진수를 사용하는 컴퓨터는 사실 0과 1밖에 없습니다.

2차원 배열의 논리적 모양은 어떻게 생겼을까요?? 아래와 같습니다.

행번호 \ 열번호					
	0	1	2	3	4
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
3	a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]
4	a[4][0]	a[4][1]	a[4][2]	a[4][3]	a[4][4]

자세히 보시면 아시겠지만 배열이름[행번호][열번호]로 배열의 각 이름이 지정됩니다. 그럼 코드를 통해 이해해보도록 하겠습니다.

코드
<pre>#include <stdio.h> int main() { char b[2][5]={'a','b','c','d','e','f','g','h','i','j'}; printf("\n\n %s", b); }</pre>

위 코드 속 배열의 모습은 아래와 같습니다.

배열b의 주소	b[0][0]	b[0][1]	b[0][2]	b[0][3]	b[0][4]
	b[1][0]	b[1][1]	b[1][2]	b[1][3]	b[1][4]
배열b의 내용	a	b	c	d	e
	f	g	h	i	j

실행시켜 볼까요?

실행을 하더라도 위에서처럼 배열의 논리적인 모양이나 어느 주소에 무엇이 들어있는지는 알 수 없습니다.

하지만 연속적인 값을 저장할 때 2차원 배열을 자주 사용하게 됩니다.

가령 이런 자료들입니다.

이름 \ 과목				
	국어	영어	수학	컴퓨터
김마장	a[0][0]	a[0][1]	a[0][2]	a[0][3]
박수원	a[1][0]	a[1][1]	a[1][2]	a[1][3]
이서울	a[2][0]	a[2][1]	a[2][2]	a[2][3]

이런 자료들을 활용하여 총점, 평균, 석차 등을 계산하기에는 2차원 배열이 편리하다고 볼 수 있습니다.

또한 웹에서 회원가입을 받을 때 특정한 사람들의 ID, 비밀번호, 주소, 전화번호 등의 개인정보를 수집하고 저장할 때에도 자주 쓰이게 됩니다.

또한 출력할 때 반복문이나 조건문을 사용하게 되면 특정한 사용자는 특정한 배열 위치만 검색할 수 있게 할 수 있기 때문에 여러 코딩의 기초가 됩니다.

3차원 배열은 2차원 배열이 여러면 겹쳐진 모양으로 설명드릴 수 있습니다. 2차원 배열을 면으로 생각하면 x축과 y축, z축이 생성되는 것입니다.

주소는 배열이름[면번호][행번호][열번호]의 순서대로 표기합니다.

3차원 배열은 수준이 높은 활용이므로 이번 과정에서는 2차원 배열만 학습하겠습니다.

그럼 중첩반복문을 통해 2차원 배열을 활용해보도록 하겠습니다.

코드

```
#include <stdio.h>
int main()
{
    int i, j;
    int a[3][4] = {{0,1,2,3},{4,5,6,7},{8,9,10,11}};

    for(i=0; i<=2; i++)
        for(j=0; j<=3; j++)
        {
            printf("%d \t", a[i][j]);
        }
}
```

중첩 반복문은 쉽게 이야기해서 반복문 속에 반복문이 들어있는 것입니다.

앞에서는 구구단을 통해 중첩 반복문을 학습했었습니다.

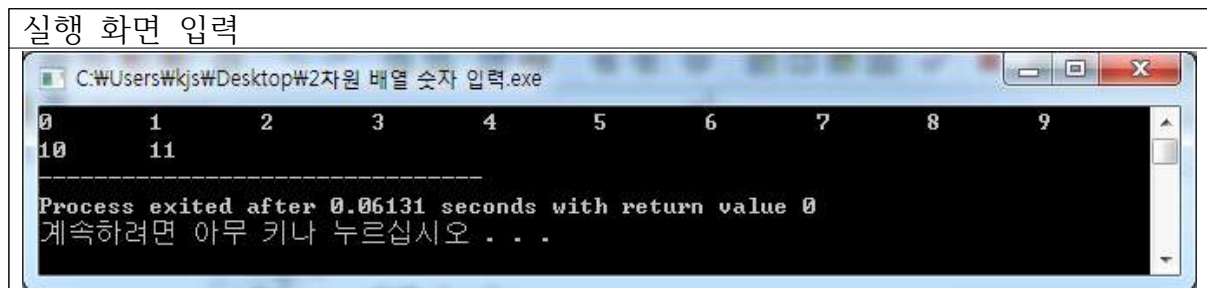
이번에는 0행에서 0열, 1열, 2열, 3열을 방문한 후, 1행으로 시작해서 다시 0열, 1열, 2열, 3열을 방문하게 됩니다. 그다음은 2행으로 시작해서 0열, 1열, 2열, 3열을 방문하고 마무리하게 됩니다.

물론 방문이라는 표현보다는 그 자리에 저장된 자료를 프린트한다고 하는 것이 더 맞겠습니다.

총 12번의 프린트를 하게 되며, \t명령을 통해 출력은 한 탭씩 떨어져서 배치됩니다.

실행화면을 보시겠습니다.

실행 화면 입력



그럼 3차원 배열은 반복문이 3번 중첩되는 구조겠죠??

이렇게 필요에 따라 알맞은 배열의 형태를 사용할 수 있습니다.

23. 혈액형 구분하기(switch - case 문 활용)

일반적으로 혈액형은 A형, B형, O형, AB형이 있습니다.

만약 이 네 가지 경우의 수를 구분하기 위해서 if()를 사용하게 된다면 3개의 if()가 필요하게 됩니다. 더 힘든 경우도 있습니다. 90점 이상은 A, 80점 이상은 B, 70점 이상은 C, 60점 이상은 D, 50점 이상은 E, 50점 미만은 F라고 한다면 이를 구분하기 위한 if()는 몇 개 필요할까요??

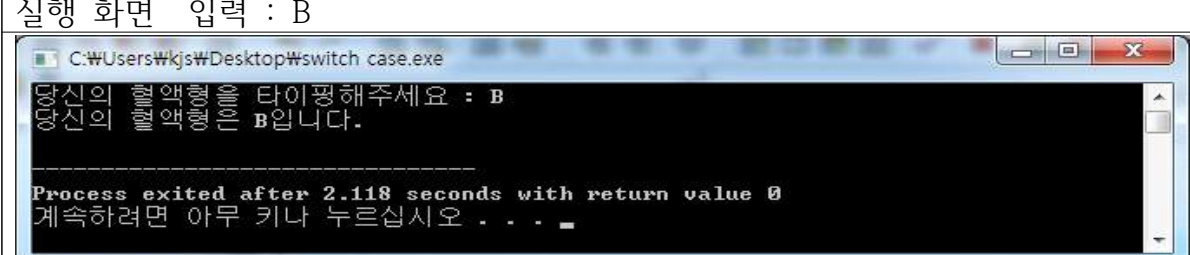
이렇게 여러 가지 조건을 다룰 때 편리한 함수가 바로 switch - case 문입니다.

먼저 실행결과 4가지를 보여드릴테니 혼자 코딩해보시겠습니까?

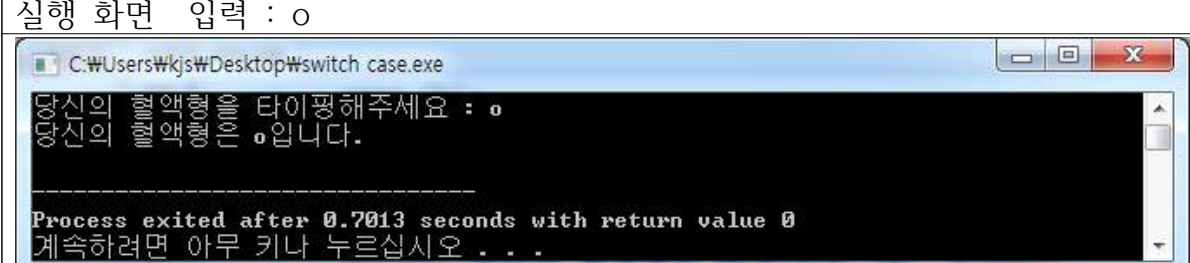
실행 화면 입력 : a



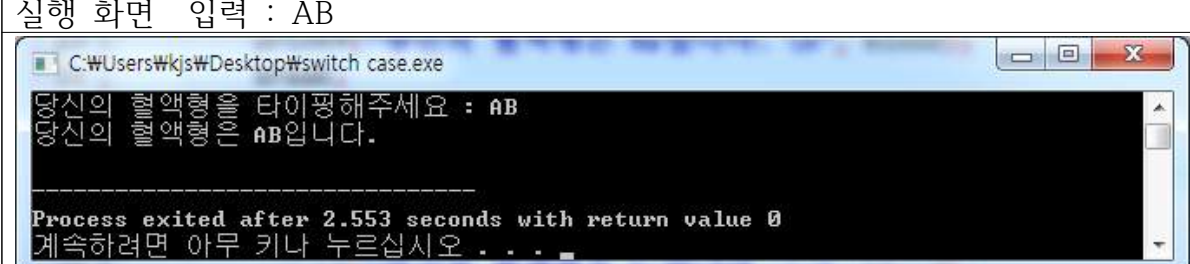
실행 화면 입력 : B



실행 화면 입력 : o



실행 화면 입력 : AB



이제 코드를 함께 작성해보겠습니다.

코드

```
#include <stdio.h>
int main()
{
    char blood[2];
    printf("당신의 혈액형을 타이핑해주세요 : ");
    scanf("\n %s", blood);

    switch(blood[0])
    {
        case 'A':
            printf("당신의 혈액형은 %s입니다. \n", blood);
            break;
        case 'a':
            printf("당신의 혈액형은 %s입니다. \n", blood);
            break;

        case 'B':
            printf("당신의 혈액형은 %s입니다. \n", blood);
            break;
        case 'b':
            printf("당신의 혈액형은 %s입니다. \n", blood);
            break;

        case 'O':
            printf("당신의 혈액형은 %s입니다. \n", blood);
            break;
        case 'o':
            printf("당신의 혈액형은 %s입니다. \n", blood);
            break;

        case 'AB':
            printf("당신의 혈액형은 %s %s입니다. \n", blood[0], blood[1]);
            break;
        case 'ab':
            printf("당신의 혈액형은 %s %s입니다. \n", blood[0], blood[1]);
            break;

    }
    return 0;
}
```


소문자를 입력하는 사람도 있을 것이고 대문자를 입력하는 사람도 있을 것이므로 여러 가지 경우를 대비한 코드입니다.

break:문은 현재의 코딩 깊이에서 한 단계 빠져나오는 것으로 switch()문에서 break:를 만나면 switch()문을 빠져나오게 되는 것입니다.

즉, 혈액형에 대한 구분을 짓게 되면 더 이상의 비교 없이 코딩이 끝나게 되는 것입니다.

위의 코드는 배열을 사용하여 AB형을 입력받았습니다. 따라서 배열에 대한 이해가 높아야 합니다. 쉽게 이야기하면 blood의 첫 번째 저장고에 A를 저장하고 두 번째 저장고에 B를 저장한 후 입력된 알파벳과 비교하는 것입니다.

스위치라는 단어를 언제 사용하시나요?

전등을 켜고 끌 때 사용하는 것을 스위치라고 하지 않습니까? 스위치는 상태를 전환시키는 뜻을 가지고 있습니다.

그렇다면 switch(blood[0])라는 것은 blood 배열 속에 들어있는 내용에 따라 상태가 변한다는 뜻이 됩니다.

blood 배열 속에 들어있는 내용에 따라 프로그램 상에 표시된 case 뒤의 문자 상수와 비교되는 것입니다.

즉, blood 배열 속에 들어있는 내용이(사용자가 입력한 혈액형) case뒤의 글자와 같으면 break:를 만나기 전까지의 명령어가 실행되는 것입니다.

이렇게 여러 가지의 경우의 수가 있을 때 switch()문을 통해 if()문을 많이 절약할 수 있기 때문에 프로그램이 간결해지는 것입니다.