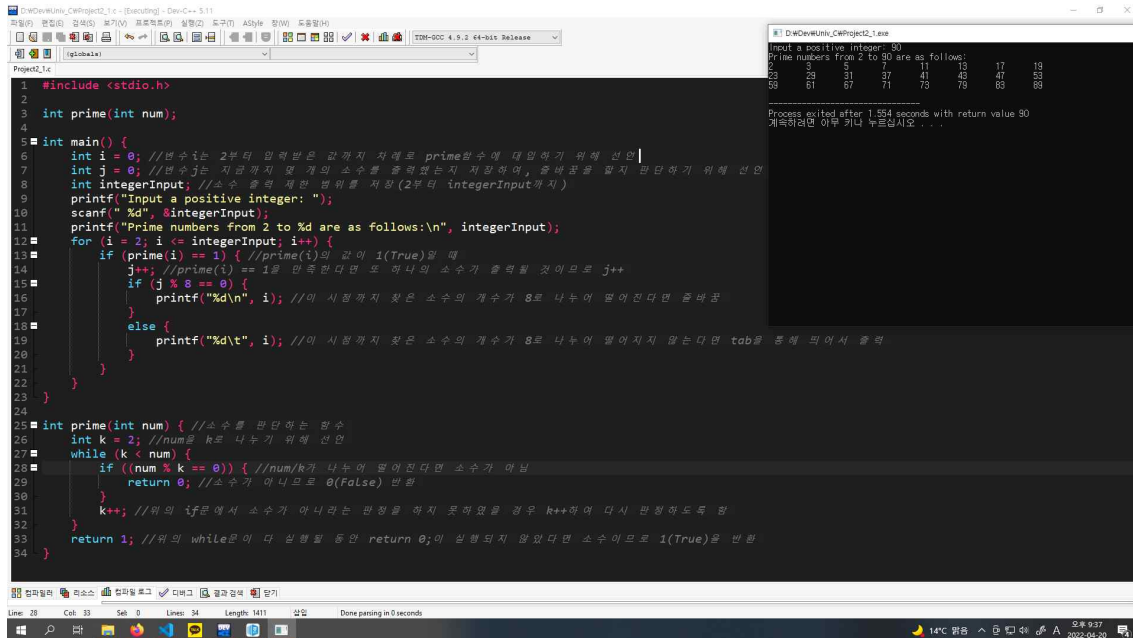


<2022-1학기 C프로그래밍 Project#2: Loops, Conditional operators, and functions 결과 보고서>

C211026 김명주

1. 범위 안의 소수를 출력하는 프로그램



```
1 #include <stdio.h>
2
3 int prime(int num);
4
5 int main() {
6     int i = 0; //변수 i는 2부터 입력받은 값까지 차례로 prime함수에 대입하기 위해 선언
7     int j = 0; //변수 j는 지금까지 몇 개의 소수를 출력했는지 저장하며, 출력문을 일지 판단하기 위해 선언
8     int integerInput; //소수 출력 범위 설정 (2부터 integerInput까지)
9     printf("Input a positive integer: ");
10    scanf("%d", &integerInput);
11    printf("Prime numbers from 2 to %d are as follows:\n", integerInput);
12    for (i = 2; i <= integerInput; i++) {
13        if (prime(i) == 1) { //prime(i)의 값이 1(True)일 때
14            j++; //prime(i) == 1을 만족한다면 또 하나의 소수가 출력될 것이므로 j++
15            if (j % 8 == 0) {
16                printf("%d\n", i); //이 시점까지 찾은 소수의 개수가 8로 나누어 떨어진다면 줄바꿈
17            }
18            else {
19                printf("%d\t", i); //이 시점까지 찾은 소수의 개수가 8로 나누어 떨어지지 않는다면 tab를 통해 띄어서 출력
20            }
21        }
22    }
23 }
24
25 int prime(int num) { //소수를 판단하는 함수
26     int k = 2; //num을 k로 나누기 위해 선언
27     while (k < num) {
28         if ((num % k == 0)) { //num/k가 나누어 떨어진다면 소수가 아님
29             return 0; //소수가 아니므로 0(False) 반환
30         }
31         k++; //위의 if문에서 소수가 아니라면 판정을 하지 못하였을 경우 k++하여 다시 판정하도록 함
32     }
33     return 1; //위의 while문이 다 실행될 동안 return 0;이 실행되지 않았다면 소수이므로 1(True)를 반환
34 }
```

Execution Output:

```
Input a positive integer: 90
Prime numbers from 2 to 90 are as follows:
2   3   5   7   11  13  17  19
23  29  31  37  41  43  47  53
59  61  67  71  73  79  83  89

Process exited after 1.554 seconds with return value 90
계속하려면 아무 키나 누르십시오 . . .
```

(1) (6th line) 변수 i는 정해진 범위(2부터 integerInput)에 해당하는 동안 prime()함수에 대입하기 위해 선언. (7th line) 변수 j는 소수의 누적 출력 개수를 저장하기 위해 선언하였다. (8th line) 변수 integerInput은 소수를 출력할 범위를 설정하기 위해 선언하였다.

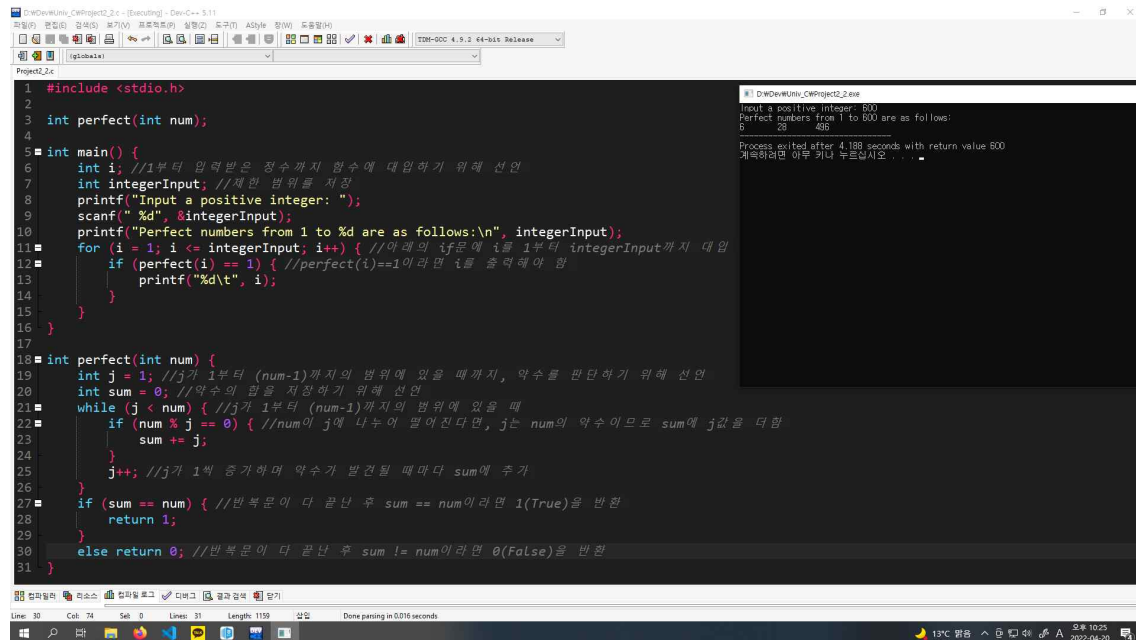
(2) (25th line~) 어떤 정수가 소수인지 판단하기 위해 prime 함수를 정의했다. k는 num을 나누기 위해 선언한 지역변수다.

(3) (27th line) while문은 k가 num보다 작을 때까지 실행하도록 했다. 사용자가 입력한 정수가 prime함수의 매개변수 num으로 들어가게 되는데, 이 정수를 2부터 (num-1)까지의 정수로 나누어 봤을 때 한 번도 나누어 떨어지지 않는다면 소수다. k가 1부터 시작한다면 num을 1로 나누는 것이므로 항상 나머지가 0이다. 소수를 판단할 때는 원래 1로 나누는 동작을 수행하지 않기 때문에 k를 2부터 시작하도록 했다.

(4) (28th line) 소수가 아닌 정수를 걸러내는 if 조건문에서, num%k==0이면 소수가 아니므로 조건에 추가하였다. if문의 조건에 당장은 해당하지 않더라도, k값을 1씩 더하여 다시 판단하면 조건에 해당하는 경우가 생길 수 있다. 그래서 k++하여 다시 while문 내부의 코드를 실행하도록 했다.

(5) (15~20th line) 소수를 한 줄에 8개씩 출력하도록 한 문장이다. 소수의 누적 출력 개수인 j가 8의 배수라면 줄바꿈을, 8의 배수가 아니라면 일정간격 띄어쓰기된다.

2. 범위 안의 완전수를 출력하는 프로그램



```
1 #include <stdio.h>
2
3 int perfect(int num);
4
5 int main() {
6     int i; //1부터 입력받은 정수까지 함수에 대입하기 위해 선언
7     int integerInput; //제한 범위를 저장
8     printf("Input a positive integer: ");
9     scanf("%d", &integerInput);
10    printf("Perfect numbers from 1 to %d are as follows:\n", integerInput);
11    for (i = 1; i <= integerInput; i++) { //각각의 i분에 i를 1부터 integerInput까지 대입
12        if (perfect(i) == 1) { //perfect(i)==1이라면 i를 출력해야 함
13            printf("%d\t", i);
14        }
15    }
16 }
17
18 int perfect(int num) {
19     int j = 1; //j가 1부터 (num-1)까지의 범위에 있을 때까지, 약수를 판단하기 위해 선언
20     int sum = 0; //약수의 합을 저장하기 위해 선언
21     while (j < num) { //j가 1부터 (num-1)까지의 범위에 있을 때
22         if (num % j == 0) { //num이 j에 나누어 떨어진다면, j는 num의 약수이므로 sum에 j값을 더함
23             sum += j;
24         }
25         j++; //j가 1씩 증가하며 약수가 발견될 때마다 sum에 추가
26     }
27     if (sum == num) { //반복문이 다 끝난 후 sum == num이라면 1(True)을 반환
28         return 1;
29     }
30     else return 0; //반복문이 다 끝난 후 sum != num이라면 0(False)을 반환
31 }
```

Output:

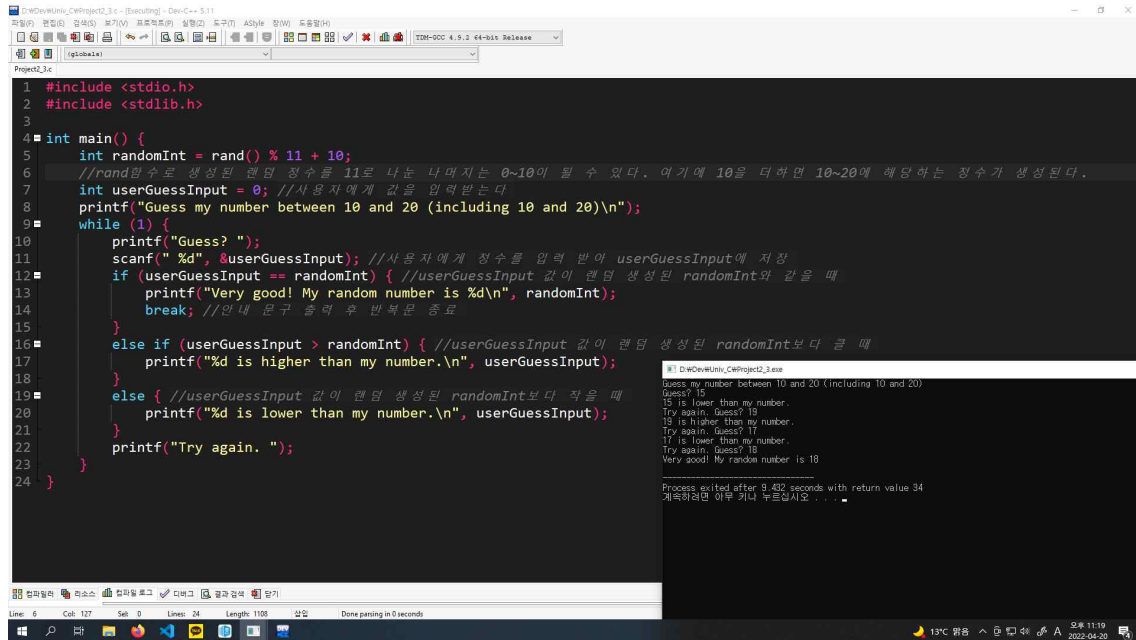
```
Input a positive integer: 600
Perfect numbers from 1 to 600 are as follows:
6      28      496
Process exited after 4.188 seconds with return value 600
계속하려면 아무 키나 누르십시오 . . .
```

(1) (21st line) j를 1부터 num까지 1씩 더하면서 num을 나누어 떨어지게 하는지 판단한다. num이 j로 나누어 떨어진다면 j는 num의 약수이다. 그러면 sum에 값을 추가한다. 그 후에는 j를 증가시키며 반복문이 실행되는데, 반복문이 종료되었을 때의 sum값은 num의 (자기 자신을 제외한) 약수의 합이다.

(2) (27th line) 약수의 합인 sum값이 원래 숫자인 num과 같다면, 이는 완전수이므로 1(True)를 반환한다. 같지 않다면 0(False)를 반환한다.

(3) (11th line) perfect함수에 어떤 값을 대입하면, 그 값이 완전수인지 아닌지 판단해준다. 그러므로 i를 1부터 제한범위 integerInput까지 반복 실행하면, 완전수일 때에만 그 i값을 출력하도록 한다. (perfect(i)==1이라면 완전수)

3. 숫자 게임 만들기



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int randomInt = rand() % 11 + 10;
6     //rand함수로 생성된 랜덤 정수를 11로 나눈 나머지는 0~10이 될 수 있다. 여기에 10을 더하면 10~20에 해당하는 정수가 생성된다.
7     int userGuessInput = 0; //사용자에게 값을 입력받는다
8     printf("Guess my number between 10 and 20 (including 10 and 20)\n");
9     while (1) {
10         printf("Guess? ");
11         scanf("%d", &userGuessInput); //사용자에게 정수를 입력 받아 userGuessInput에 저장
12         if (userGuessInput == randomInt) { //userGuessInput 값이 랜덤 생성된 randomInt와 같을 때
13             printf("Very good! My random number is %d\n", randomInt);
14             break; //내 문구 출력 후 반복문 종료
15         }
16         else if (userGuessInput > randomInt) { //userGuessInput 값이 랜덤 생성된 randomInt보다 클 때
17             printf("%d is higher than my number.\n", userGuessInput);
18         }
19         else { //userGuessInput 값이 랜덤 생성된 randomInt보다 작을 때
20             printf("%d is lower than my number.\n", userGuessInput);
21         }
22         printf("Try again. ");
23     }
24 }
```

Output:

```
Guess my number between 10 and 20 (including 10 and 20)
Guess? 15
15 is lower than my number.
Try again. Guess? 18
18 is higher than my number.
Try again. Guess? 17
17 is lower than my number.
Try again. Guess? 18
Very good! My random number is 18

Process exited after 9.432 seconds with return value 34
계속하려면 아무 키나 누르십시오 . . .
```

(1) (Project2 안내문에서 참고한 부분) 랜덤 정수를 생성하기 위해 `#include <stdlib.h>`을 추가했다. 또한 `rand`함수는 `[0~32767]` 사이의 정수값을 반환하고, 어떤 수를 10으로 나눈 나머지는 `0~9`사이의 정수값이라는 점을 참고했다. 이 프로그램에서는 `[10, 20]` 사이의 정수를 랜덤으로 생성해야 한다. 일단 `[0, 10]` 사이의 정수를 생성하도록 한 후, 10을 더하면 `[10, 20]` 사이의 정수가 된다. 그래서 5번째 줄과 같이 코드를 작성했다.

(2) (9th line) `while`문은 사용자가 정답을 입력할 때까지 계속 반복된다. 랜덤 숫자와 입력값이 같다면 `break`문을 통해 반복문에서 빠져나온다. 입력값이 틀렸다면 상황에 맞는(대소비교) 안내 문구를 출력한 후 `while`이 다시 반복된다.

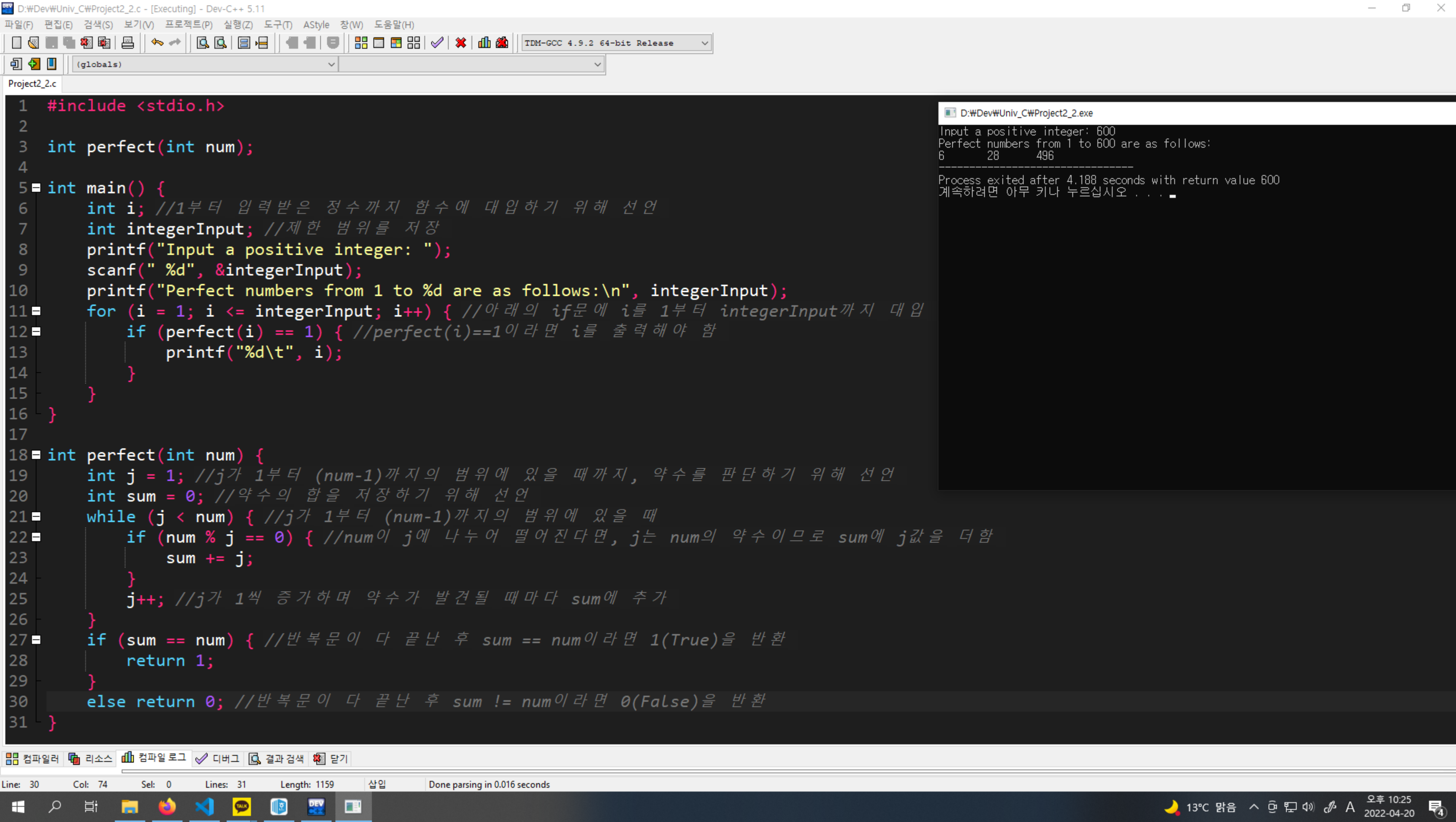
Project2_1.c

```
1 #include <stdio.h>
2
3 int prime(int num);
4
5 int main() {
6     int i = 0; //변수 i는 2부터 입력받은 값까지 차례로 prime함수에 대입하기 위해 선언
7     int j = 0; //변수 j는 지금까지 몇 개의 소수를 출력했는지 저장하여, 줄바꿈을 할지 판단하기 위해 선언
8     int integerInput; //소수 출력 제한 범위를 저장 (2부터 integerInput까지)
9     printf("Input a positive integer: ");
10    scanf("%d", &integerInput);
11    printf("Prime numbers from 2 to %d are as follows:\n", integerInput);
12    for (i = 2; i <= integerInput; i++) {
13        if (prime(i) == 1) { //prime(i)의 값이 1(True)일 때
14            j++; //prime(i) == 1을 만족한다면 또 하나의 소수가 출력될 것이므로 j++
15            if (j % 8 == 0) {
16                printf("%d\n", i); //이 시점까지 찾은 소수의 개수가 8로 나누어 떨어진다면 줄바꿈
17            }
18            else {
19                printf("%d\t", i); //이 시점까지 찾은 소수의 개수가 8로 나누어 떨어지지 않는다면 tab을 통해 띄어서 출력
20            }
21        }
22    }
23 }
24
25 int prime(int num) { //소수를 판단하는 함수
26     int k = 2; //num을 k로 나누기 위해 선언
27     while (k < num) {
28         if ((num % k == 0)) { //num/k가 나누어 떨어진다면 소수가 아님
29             return 0; //소수가 아니므로 0(False) 반환
30         }
31         k++; //위의 if문에서 소수가 아니라는 판정을 하지 못하였을 경우 k++하여 다시 판정하도록 함
32     }
33     return 1; //위의 while문이 다 실행될 동안 return 0;이 실행되지 않았다면 소수이므로 1(True)을 반환
34 }
```

D:\Dev\Univ_C\Project2_1.exe

```
Input a positive integer: 90
Prime numbers from 2 to 90 are as follows:
2      3      5      7      11     13     17     19
23     29     31     37     41     43     47     53
59     61     67     71     73     79     83     89
```

```
-----
Process exited after 1.554 seconds with return value 90
계속하려면 아무 키나 누르십시오 . . .
```




```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int randomInt = rand() % 11 + 10;
6     //rand함수로 생성된 랜덤 정수를 11로 나눈 나머지는 0~10이 될 수 있다. 여기에 10을 더하면 10~20에 해당하는 정수가 생성된다.
7     int userGuessInput = 0; //사용자에게 값을 입력받는다
8     printf("Guess my number between 10 and 20 (including 10 and 20)\n");
9     while (1) {
10         printf("Guess? ");
11         scanf(" %d", &userGuessInput); //사용자에게 정수를 입력받아 userGuessInput에 저장
12         if (userGuessInput == randomInt) { //userGuessInput 값이 랜덤 생성된 randomInt와 같을 때
13             printf("Very good! My random number is %d\n", randomInt);
14             break; //안내 문구 출력 후 반복문 종료
15         }
16         else if (userGuessInput > randomInt) { //userGuessInput 값이 랜덤 생성된 randomInt보다 클 때
17             printf("%d is higher than my number.\n", userGuessInput);
18         }
19         else { //userGuessInput 값이 랜덤 생성된 randomInt보다 작을 때
20             printf("%d is lower than my number.\n", userGuessInput);
21         }
22         printf("Try again. ");
23     }
24 }
```

D:\Dev\Univ_C\Project2_3.exe

```
Guess my number between 10 and 20 (including 10 and 20)
Guess? 15
15 is lower than my number.
Try again. Guess? 19
19 is higher than my number.
Try again. Guess? 17
17 is lower than my number.
Try again. Guess? 18
Very good! My random number is 18
```

```
-----
Process exited after 9.432 seconds with return value 34
계속하려면 아무 키나 누르십시오 . . .
```