**Report 2 – 설명**

**20182931 신석경**

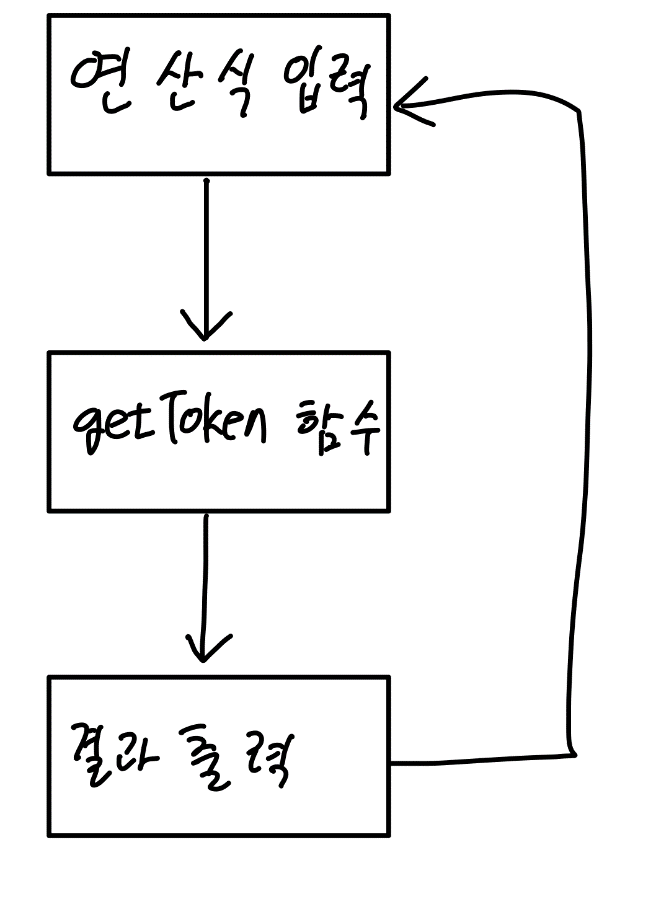
Report 2-1. 연속된 연산식에서 각 연산이 몇 번 나오는 지 출력하는 프로그램

이 프로그램은 반복적으로 연산식을 입력 받고, 그 속에 있는 연산 부호(+, -, \*, /, %)의 개수를 카운트하여 출력하는 프로그램이다. 이 프로그램의 메인 흐름은

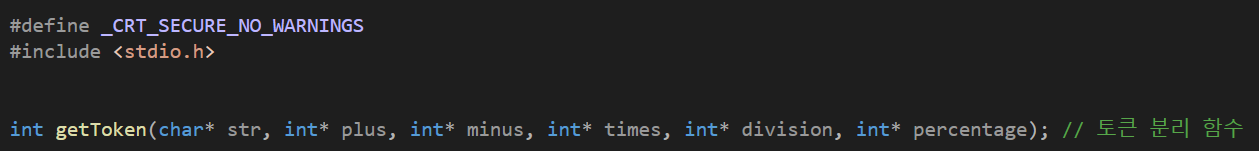
1번. 연산식을 입력 받는다.

2번. 함수를 통해 연산 부호의 개수를 update한다.

3번. 결과를 출력하고 다시 1번 루프로 돌아온다.

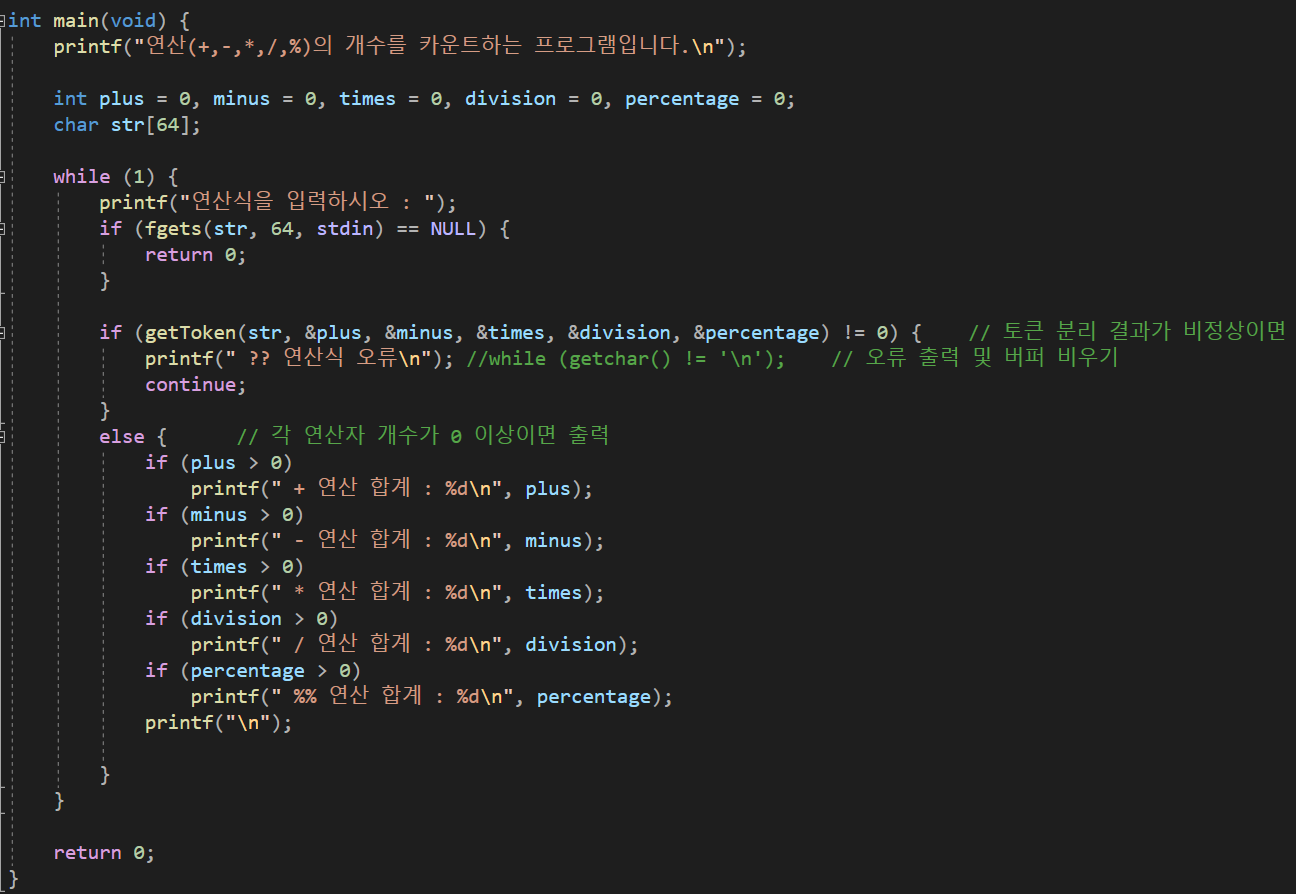
프로그램의 동작 흐름을 살펴보면 옆의 그림과 같다. 연산식을 입력 받고 getToken 함수를 통해 연산식을 검사한다. 그 이후, 함수가 정상적으로 종료되었으면 각 연산자의 개수를 출력하고, 비정상 종료이면 오류를 출력하고 다시 연산식을 입력 받는다.

이 프로그램은 반복적 루프이기 때문에 while문을 사용하였다. While 문 내부에서 연산식을 입력 받은 것을 getToken 함수로 검사하고 이것의 결과에 따라 if문을 통해 오류를 출력, 혹은 연산자 개수 출력을 하게 된다. 자세하게 코드를 보며 알아보자.

먼저, 프로그램에 필요한 함수를 정의하였다. 입력된 문자열을 가지고 연산자의 개수를 계산해서 각각의 변수에 저장하는 함수이다.



함수의 구현부이다. 문자열 char\* str을 입력 받아 문자열 내부를 검사한다. 0 ~ 9, 공백, 탭 문자 일 경우 아무것도 하지 않는다. +, -, \*, /, % 일 경우에는 각각 plus\_temp, minus\_temp, times\_temp, division\_temp, percentage\_temp 에 해당하는 값을 1 씩 증가시킨다. 이렇게 temp 변수로 증가시키는 이유는 검사하다가 연산식이 잘못된 경우 이에 해당하는 연산자는 증가시키면 안되기 때문이다. 그래서 while문이 끝난 경우 즉, 정상적인 연산식인 경우 지금까지 계산된 temp 변수들을 main 문에서 선언된 각각의 변수의 값에 더해 main 문의 연산자 개수를 증가시킨다. 만약 올바르지 않은 입력이면 -1을, 문자열을 끝까지 검사했으면 0을 반환한다.

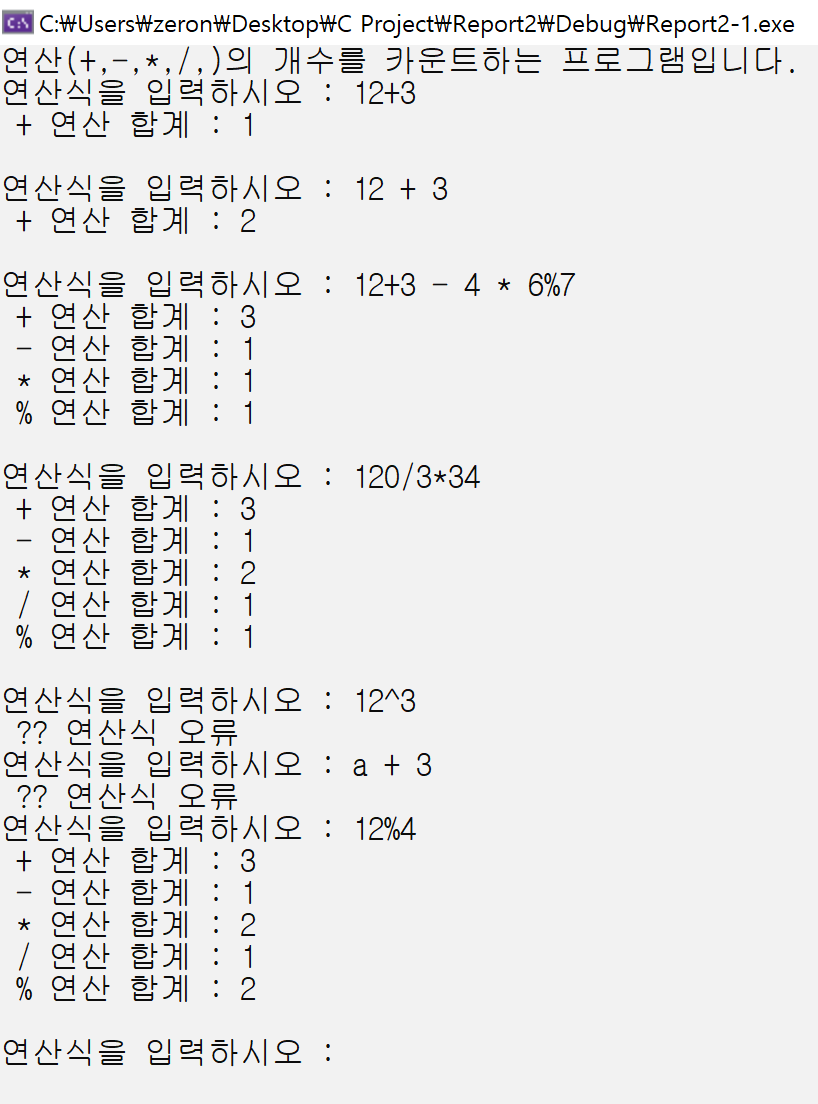


메인 함수이다. 먼저 각각의 연산자 개수를 저장하려는 변수 plus, minus, times, division, percentage를 0으로 초기화한다. 그리고 문자열을 입력 받을 str 변수도 선언한다.

while문 내부에서 str을 입력 받고, getToken에서 연산자 개수를 검사한다. 이때, 메인 문에 선언된 plus ~ percentage 변수를 update 하기 위하여 각 변수의 주소값을 argument로 넘겨준다. getToken()은 정상적으로 끝나면 0을 반환하고 비정상적이면 -1을 반환한다. 그래서 if 문에서 함수의 반환값이 0이 아니면 오류를 출력하고, 버퍼를 비우고 다시 연산식을 입력 받도록 하였다. 그리고 함수의 반환값이 0이면, 각 변수의 count 값이 0보다 큰 경우에 각 연산의 개수를 출력한다.

Str을 입력 받을 때, 공백도 입력 받기 위해 fgets로 입력 받는데, Ctrl c를 입력할 때 fgets == NULL이기 때문에 이 경우 프로그램이 정상적으로 종료되도록 하였다.

이제 각각의 상황에 대한 결과를 보겠다.



1번 출력은 공백 없이 연산자만 입력한 경우이다. 2번 출력은 공백까지 처리한 경우이다. 위의 두 경우는 ‘+’ 연산만 있기 때문에 + 연산자의 개수만 출력한다.

3번 출력은 공백, -, \*, % 모든 것을 포함한 경우이다. 현재 + 연산자의 개수가 2개이기 때문에, 이 연산식을 거치면 + 연산자의 개수가 +1 되어서 3이 되어야 한다.

4번 출력 또한 기존의 연산자 개수에 새로 들어온 연산자의 개수를 더한다.

5 ~ 6번 출력은 오류 출력이다. ‘^’ 연산자가 정의되지 않기 때문에 오류 뜨고, a는 숫자가 아니기 때문에 오류가 출력된다.

7번 출력은 정상적인 출력인데, 6번 출력에서 + 연산자가 1개 있었지만 올바르지 않았기 때문에 연산자 개수가 증가하지 않았다.

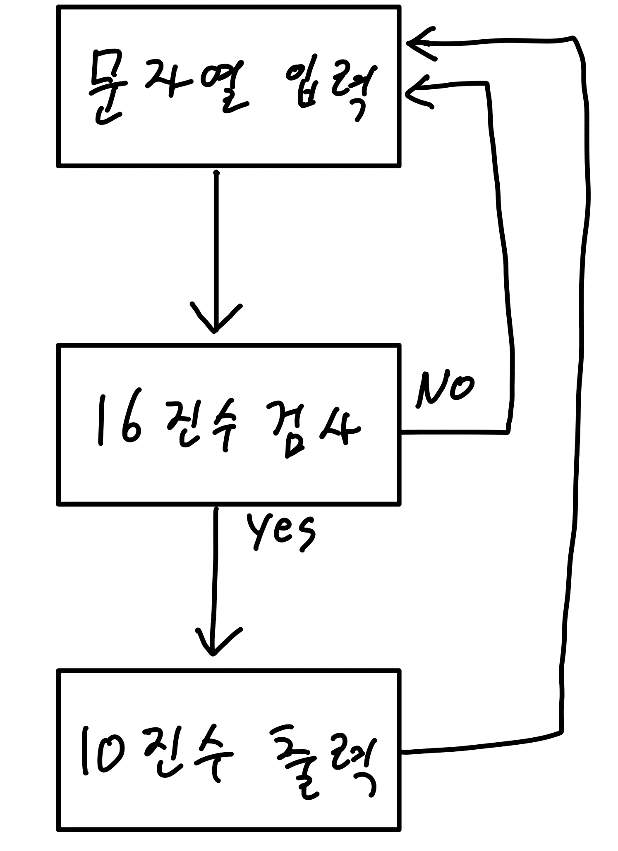
Report 2-2. 문자열로 받은 16 진수가 문자열인지 검사하고, 10진수로 변경하여 출력하는 프로그램

이 프로그램은 반복적으로 16진수를 문자열로 입력 받아, 올바른 16 진수인지 확인한 다음 이 값을 10진수로 바꿔서 출력한다. 이 프로그램의 흐름은

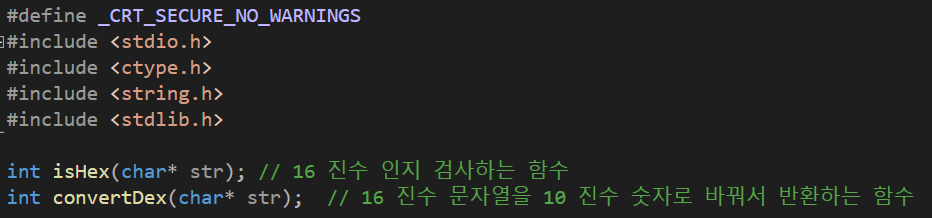
1번. 16 진수를 문자열로 입력 받는다.

2번. 입력된 문자열이 16 진수에 해당하는지 확인한다.

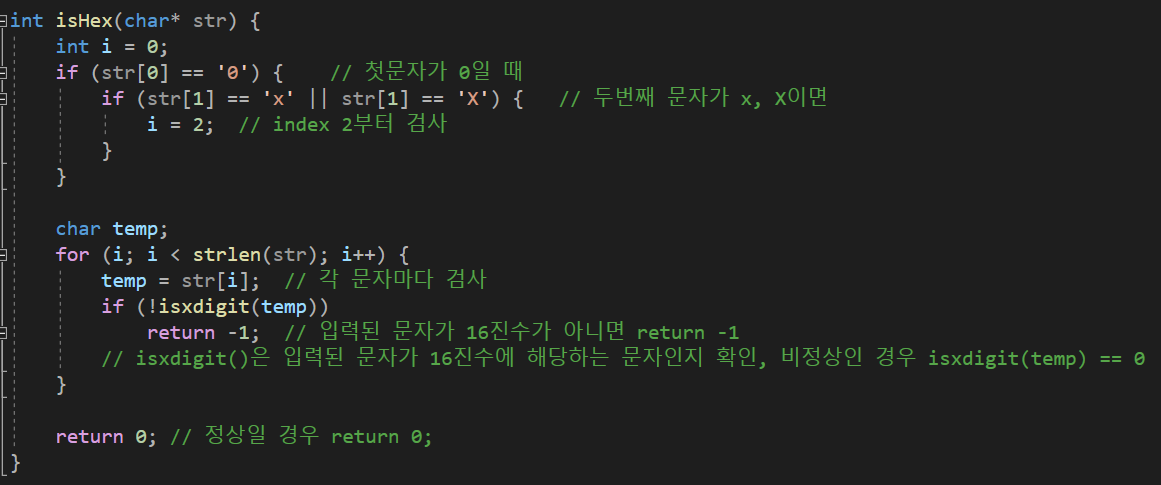
3번. 올바른 입력인 경우 10 진수로 바꿔서 출력하고 다시 1번으로 돌아간다.

 프로그램의 흐름 그림이다. 문자열을 입력 받고 16 진수 문자열인지 검사한다. 16 진수가 아니면 다시 문자열을 입력 받고, 16 진수가 맞으면 10진수로 출력하고 다시 문자열을 입력 받는다.

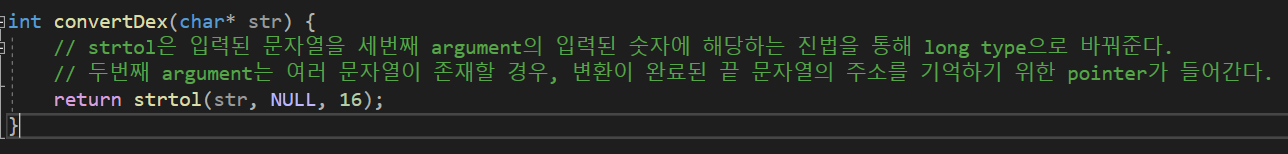
이 프로그램은 일정 패턴의 반복 실행이기 때문에 while문을 사용한다. 문자열이 16 진수인지 확인하기 위하여 isHex함수를 구현하고 그 내부에서 ctype.h 헤더에 포함되어 있는 isxdigit 함수를 사용하여 16 진수를 검사한다. 또한, 10진수로 바꾸기 위하여 convertDex 함수를 구현하였는데, 내부에서는 string.h에 저장되어 있는 strtol 함수를 사용하여 10 진수로 바꾼다. 자세한 것은 코드를 보며 알아보자.



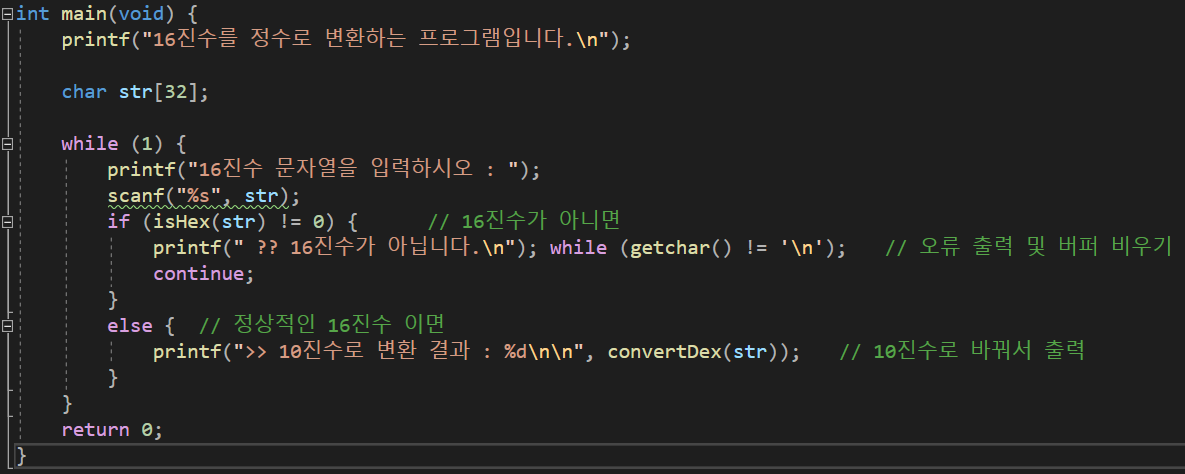
먼저, 정의된 함수이다. 입력된 문자열이 16 진수인지 확인하기 위해 isHex 함수를 정의하였고, 10 진수로 반환하는 함수로 convertDex 함수를 정의하였다. 그 내부에서 사용되는 isxdigit 함수를 사용하기 위해 ctype.h를, strtol 함수를 사용하기 위해 stdlib.h를, strlen을 사용하기 위해 string.h를 include 하였다.



16 진수인지 확인하는 isHex 함수이다. 입력된 문자열이 0x 혹은 0X로 시작될 가능성이 있기 때문에 그것을 먼저 확인한다. 0번 1번 인덱스가 각각 ‘0’, ‘x’ or ‘X’ 이면 시작 index를 2로 바꾼다. 이후 str의 문자 하나하나씩 isxdigit을 통해 16진수에 해당하는 문자인지 검사한다. 만약 아닐 경우 -1을 return하고 정상적으로 끝난 경우 0을 반환한다.



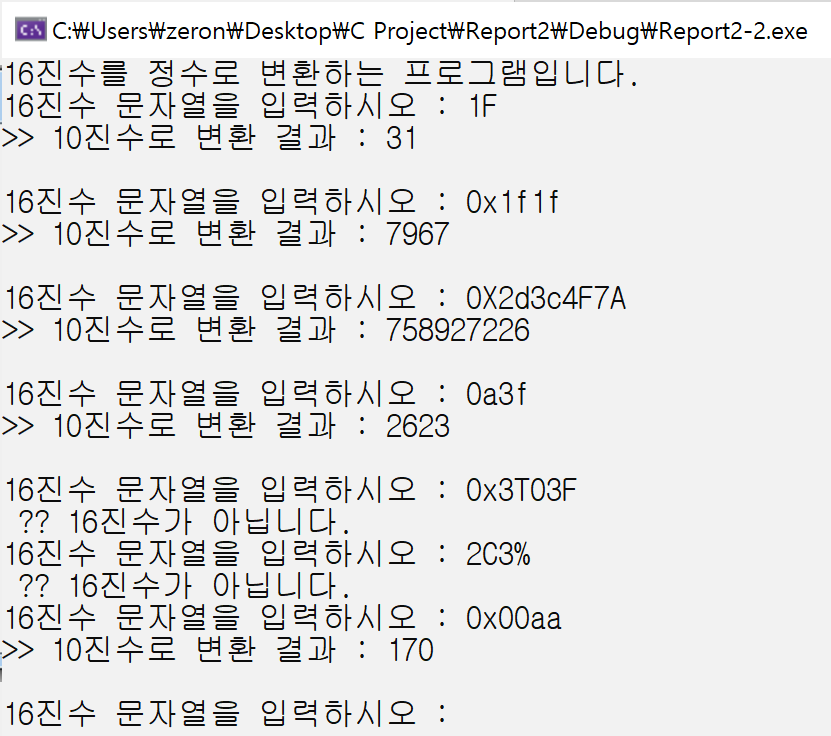
convertDex 내부의 strtol(str, NULL, 16)을 통해 16 진수를 10 진수로 반환한다. strtol 함수는 string to long 인 함수로 문자열을 long type으로 바꾸는 함수이다. 첫번째 argument는 바꿀 string이다. 세번째 argument는 어떤 진법을 통해 long type로 바꿀 지 정하는 것이다. 이 문제의 경우 16진수이기 때문에 16을 넘어간다. 두번째 argument는 변환이 끝난 주소값이다. 예를 들어, char\* str = “1f 6f 25f” 인 경우, strtol(str, &end, 16)인 경우 1f를 10진수로 바꾼 31을 반환하고 end에는 6f 문자열이 시작하는 첫 주소값이 들어간다. 우리의 경우 문자열 1개만 들어오기 때문에 NULL을 넘긴다.



메인 함수이다. 먼저 문자열을 입력 받을 str변수를 선언한다. 먼저, 문자열을 입력 받고 isHex(str)을 통해 16 진수인지 확인한다. isHex(str)이 0이 아닌 경우 16 진수가 아니기 때문에 오류 출력과 버퍼를 비우고 다시 입력을 받는다.

만약 isHex(str)이 0인 경우, convertDex(str)을 통해 10진수로 바꾸어 출력하고 다시 입력을 받는다.

이에 대한 출력을 보자.



1 ~ 4 출력을 보면 0x가 있던 없던 16진수를 10진수로 올바르게 변환한 것을 알 수 있다. 5 ~ 6 출력을 보면 16 진수 문자에 포함되어 있지 않은 ‘T’가 있기 때문에 오류를 출력하고, ‘%’ 문자가 포함되어 있기 때문에 오류를 출력한다. 이후 다시 정상적인 입력이 들어오면 10진수로 바꾸어 출력한다.

Report 2-3. 일상 생활에서 다양한 암호를 사용한다. 사용자로부터 암호를 입력받고, 암호의 안전도를 검사하는 프로그램

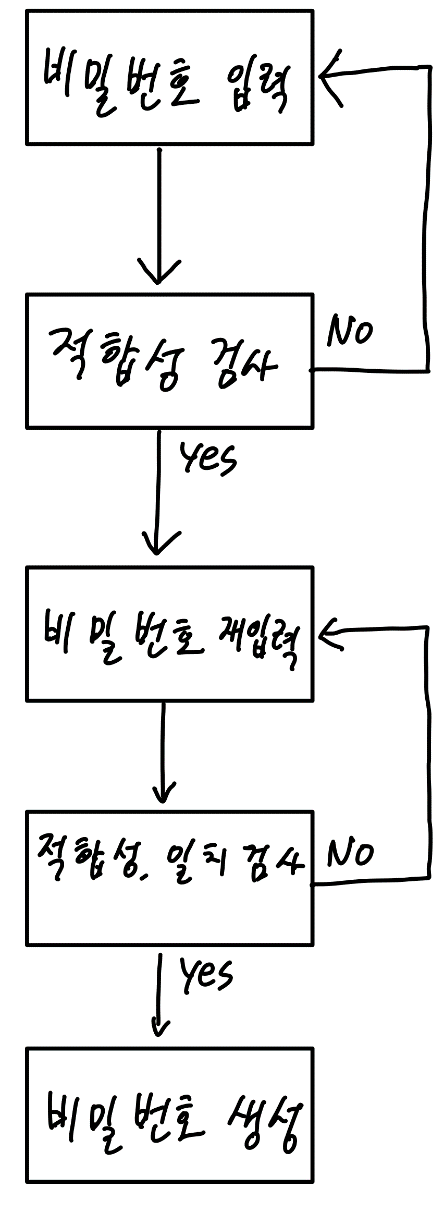
이 프로그램은 사용자의 입력을 분석하여 조건에 맞으면 비밀번호를 생성하고, 다시 재입력하여 확인하는 프로그램이다. 이 조건은 대문자, 소문자, 숫자, 그리고 그외의 문자(특수문자, 기호) 4가지 중 3가지 이상 포함하고, 8글자 이상, 12글자 이하여야 한다. 이 프로그램의 흐름은

1번. 비밀번호를 입력한다.

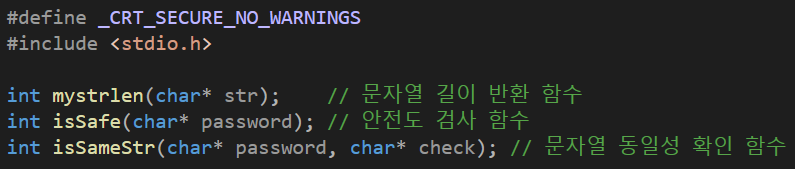
2번. 적합성을 검사하고 No이면 1번으로, Yes이면 다음으로 넘어간다.

3번. 비밀번호를 재입력한다.

4번. 일치성을 검사하고 No이면 3번으로, Yes이면 정상 종료한다.

 프로그램의 흐름을 그림으로 나타낸 것이다. 비밀번호를 입력 받고 적합성을 검사한다. 이 결과에 따라 다시 입력 받거나, 생성하고 확인을 위하여 재입력을 받는다.이후 다시 재입력 받아서 생성된 비밀번호와 일치성을 검사하고 맞으면 정상 종료되고 아닌 경우 다시 재입력 받는다.

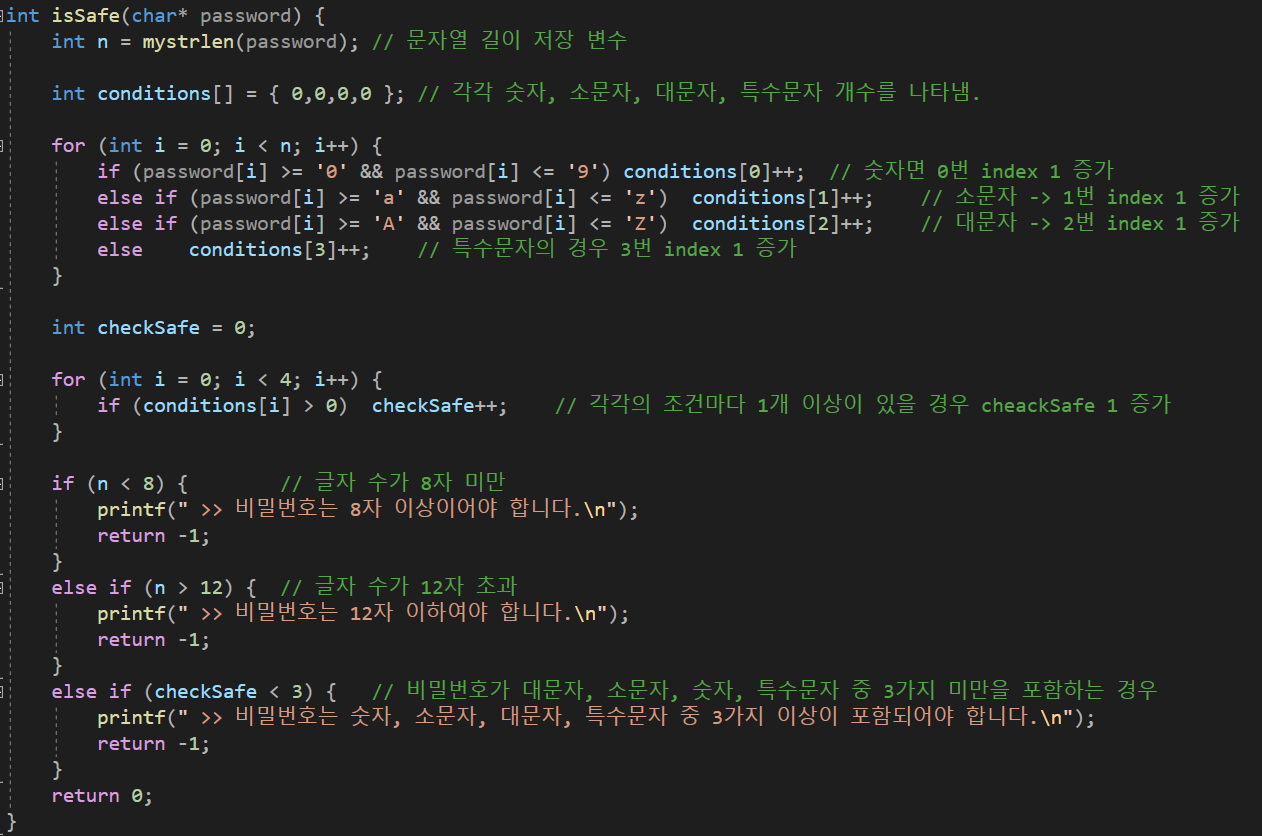
흐름도를 살펴보면 2가지의 loop가 필요한다. 비밀번호를 생성하는 부분, 일치성을 검사하는 부분이다. 그리고 적합성을 검사하는 함수와, 문자열 일치를 검사하는 함수를 사용한다.



정의된 함수로는 3개가 있다. 먼저 문자열 길이를 구하는 mystrlen 함수이다. 이는 여러 문자열 관련 연산을 할 때 사용한다. 그리고 적합성을 확인하는 isSafe 함수, 두 문자열의 일치를 확인하는 isSameStr 함수이다.

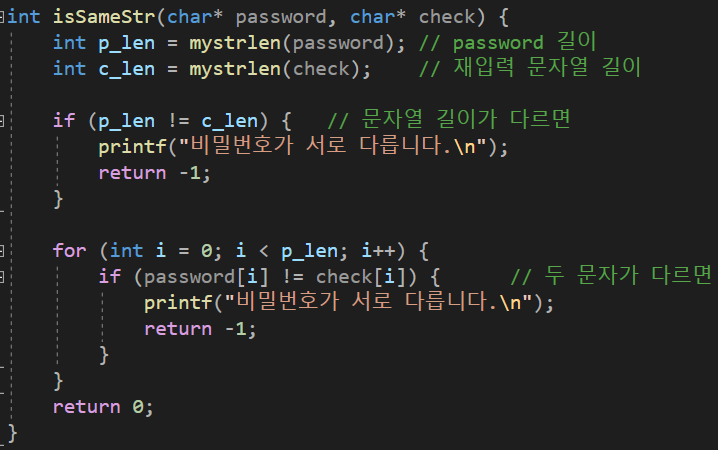


변수 n을 정의하고, str[n]이 존재하는 경우 n을 증가한다. 그리고 존재하지 않으면 n을 return 한다.



적합성을 판단하는 함수이다. 먼저 password 문자열 길이를 구해 n에 저장한다. 그리고 대문자, 소문자, 숫자, 그리고 그 외의 문자(특수문자, 기호) 4가지에 해당하는 개수를 저장하기 위해 int conditions[]을 {0,0,0,0}로 초기화한다. 이후 문자열 길이 안에서 검사하면서 조건에 맞는 index에 해당하는 값을 1 증가한다. 전체를 다 검사한 이후 4가지 조건 중 3가지 이상인지 확인하기 위하여 checkSafe = 0 을 선언하고 conditions 배열을 검사하며 각 값이 0보다 큰 경우 checkSafe 값을 1씩 증가시킨다.

이후, 문자열 길이가 8 미만, 12 초과 인 경우 오류를 출력, -1을 return하고, checkSafe가 3 미만인 경우 4가지 중 3가지 이상을 만족하지 않기 때문에 오류 출력, -1을 return한다.



두 문자열이 동일한지 검사하는 함수이다. 먼저 두 문자열의 길이를 구해 각각 p\_len과 c\_len에 저장한다.

두 변수의 값이 다르면 애초에 두 문자열의 길이가 다르기 때문에 오류 출력, -1을 return한다.

두 변수가 같으면 이제 문자 하나씩을 검사하여 다를 경우 오류 출력, -1을 반환한다.

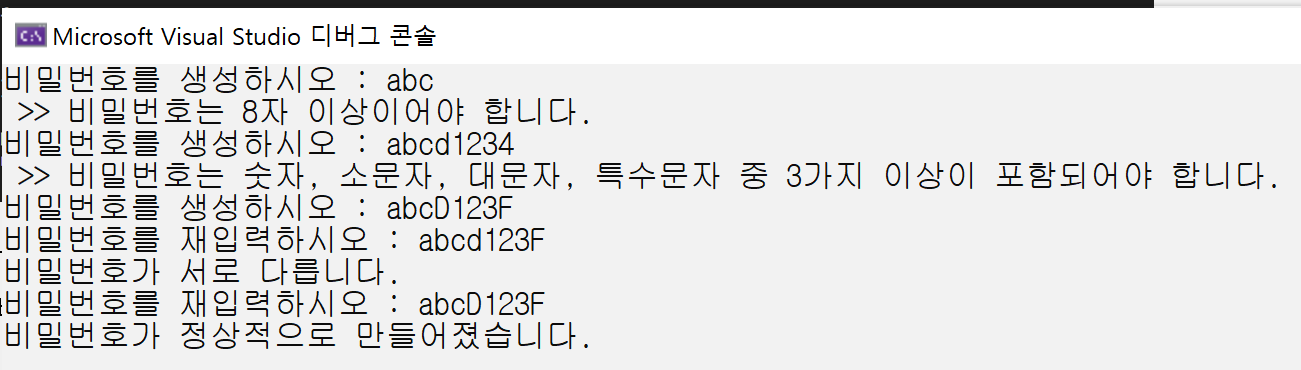
두 문자열이 같으면 0을 반환한다.



먼저 비밀번호를 입력 받을 변수 password를 선언한다. 그리고 do while 문을 통해 입력된 비밀번호가 적합한지 판단하고 적합하지 않으면 다시 입력을 받는다. 적합한 경우 isSafe가 0을 반환하기 때문에 do while 문을 정상적으로 빠져나오고 아닌 경우 다시 do while문을 실행한다.

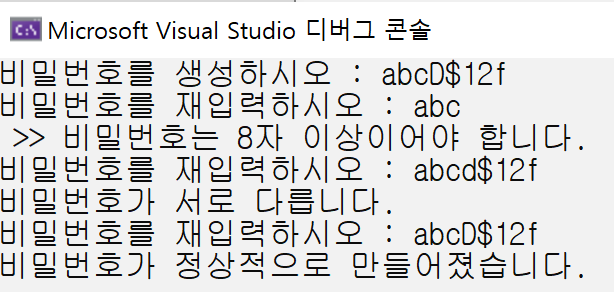
이후, 재입력 받을 변수 check를 선언하고 입력 받는다. 여기서는 두가지 검사를 하여야 하는데, 적합성 검사와 문자열 일치 검사이다. 먼저 적합성 검사를 하여 재입력된 비밀번호가 적합한지 확인하고 그 이후 문자열이 일치하는지를 검사한다. 두 함수의 반환값이 모두 0일 경우, 즉 모두 정상적으로 종료된 경우에만 do while문을 빠져나와 프로그램이 정상 종료된다.

이에 대한 출력을 알아보자.



1번 입력은 abc로 8자 미만이기 때문에 오류를 출력하고 다시 입력 받는다. 2번 입력은 소문자, 숫자로만 이루어져 있기 때문에 4가지 조건 중 2가지만 포함되어 있어 오류 출력 및 다시 입력 받는다.

3번 입력은 정상적인 비밀번호여서 확인을 위하여 재입력 받는데, 두 비밀번호가 서로 다르기 때문에 다시 입력 받고 같은 경우 프로그램을 정상 종료하였다.



이 경우 첫 비밀번호 생성은 잘 되었지만, 이후 재입력 받는 곳에서 입력된 비밀번호가 8자 미만이기 때문에 오류 출력 및 재입력, 비밀번호가 달라서 오류 출력 및 재입력을 한다. 이후 동일한 경우 정상 종료하였다.

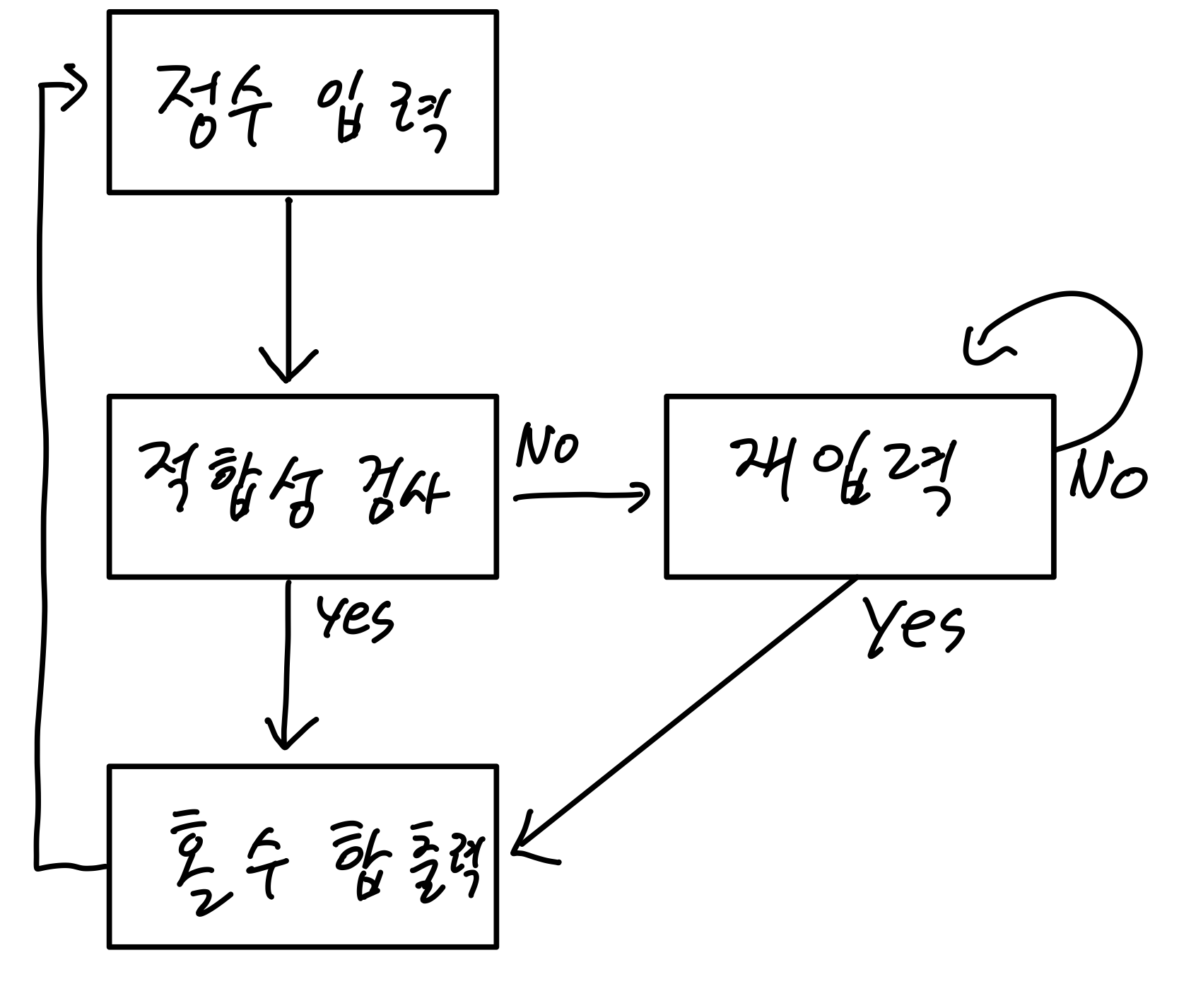
Report 2-4. 표준 입력으로 받은 정수 n에 대하여 1부터 n까지의 홀수의 합을 구하는 함수를 재귀함수로 작성하고, 합이 출력되도록 프로그램

이 프로그램은 반복적으로 정수를 입력 받고, 그에 따라 1부터 그 정수까지의 홀수의 합을 구하는 프로그램이다. 특히, 여기서는 재귀함수를 사용하여 합을 구해야 한다. 이 프로그램의 흐름은

1번. 정수를 입력 받는다.

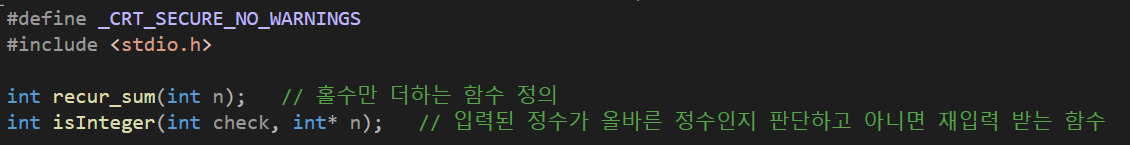
2번. 정수의 적합성을 검사한다. 적합하지 않으면 1번, 적합하면 그대로 진행한다.

3번. 결과를 출력하고 1번으로 돌아온다.

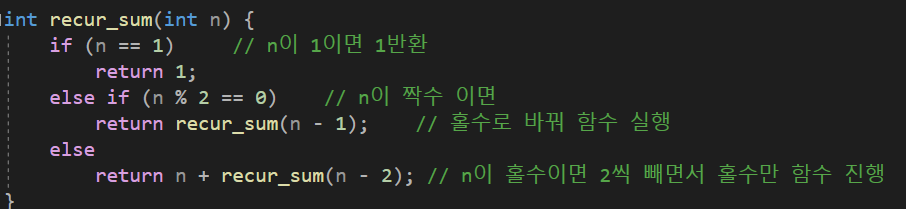


프로그램의 흐름을 그림으로 나타낸 것이다. 먼저 정수를 입력 받는다. 적합성을 검사하여 적합한 경우 홀수 합을 출력한다. 만약 적합하지 않은 경우, 계속 재입력을 받아 적합할 때까지 입력 받고, 적합한 경우 홀수 합을 출력하고 다시 처음으로 돌아간다.

이 흐름을 구현하기 위해 while문이 사용되었다. 그리고 홀수의 합을 재귀적으로 구하기위해 함수를 구현하였다. 자세한 것은 코드를 보며 알아보자.



먼저, 재귀적으로 호출할 recur\_sum 함수를 정의하였다. 또한, 올바른 정수인지 확인하고 아닐 경우 계속 재입력 받는 isInterger 함수를 정의하였다.

 recur\_sum 함수의 구현 부분이다. 정수 n을 입력 받는다. 이때, n이 1이면 1을 반환하고, n이 짝수인 경우는 recur\_sum(n-1)을 반환한다. 그리고 홀수는 n + recur\_sum(n-2)를 반환한다. 재귀적인 부분에 대해 더 설명하면, 홀수의 정수를 가지고 실행하는 함수는 그 정수와 그 이전 홀수의 recur\_sum의 결과를 더한다.

만약 n = 10이라고 해보자. 그러면 처음 시작은 recur\_sum(10)일 것이다. 그러면 이때는 짝수 이므로 recur\_sum(9)가 최종 결과가 된다.

recur\_sum(9)는 홀수 이므로 n + recur\_sum(n-2) 즉, 9 + recur\_sum(7)을 반환하고 이것이 최종 결과가 된다.

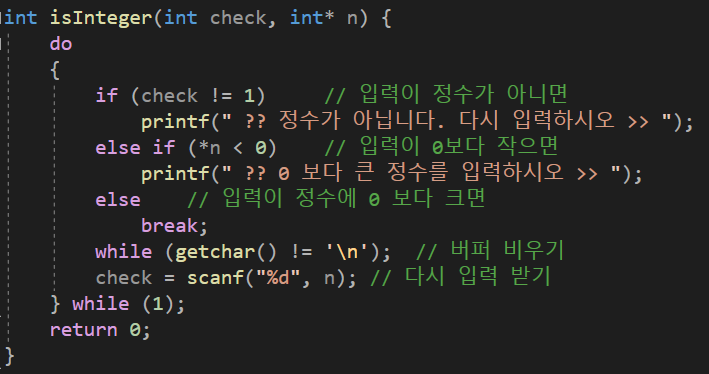
recur\_sum(7) = 7 + recur\_sum(5)

recur\_sum(5) = 5 + recur\_sum(3)

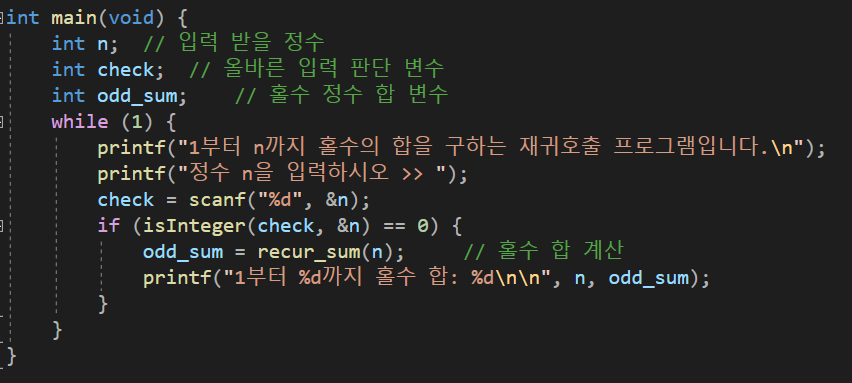
recur\_sum(3) = 3 + recur\_sum(1)

recur\_sum(1) = 1이기 때문에

recur\_sum(10) = recur\_sum(9) = 9 + recur\_sum(7) = 9 + 7 + recur\_sum(5) = 9 + 7 + 5 + recur\_sum(3) = 9 + 7 + 5 + 3 + recur\_sum(1) = 9 + 7 + 5 + 3 + 1 = 25 이 된다.

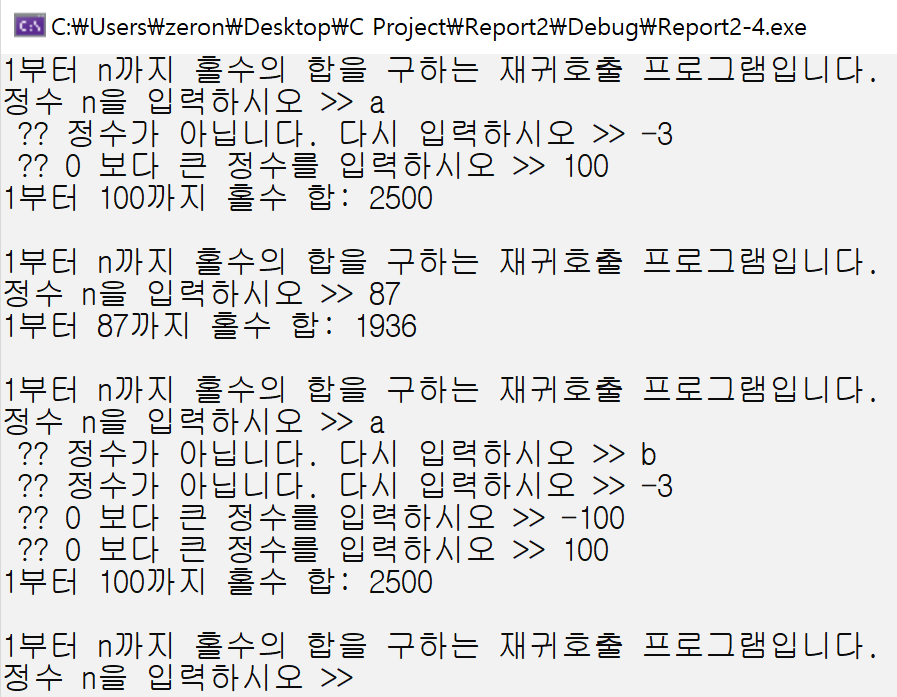


입력된 정수의 적합성을 판단하는 isInteger함수이다. 이 함수는 main 함수에서의 scanf 결과를 저장하는 check와 입력 받은 변수의 주소를 argument로 가져온다. 먼저, check가 1이 아닌 경우는 입력이 제대로 된 경우가 아니기 때문에 오류를 출력한다. 마찬가지로 \*n, 즉 입력 받은 정수의 값이 0보다 작으면 오류를 출력한다. 만약 그 두 조건 다 하지 않는 경우는 정상적인 입력으로 break를 통해 while문을 빠져나온다. 오류가 나온 경우는 버퍼를 비우고, 다시 입력을 받는다. 이때, n은 int 형 포인터이기 때문에 main 함수에서의 n을 의미한다. 즉 이 함수에서 입력을 받아도 main 함수의 n 값이 변경된다. 다시 정수를 입력 받고 check, 음수 n을 검사하여 정상적일 경우 0을 반환한다.



메인 함수이다. 먼저 입력 받을 정수를 저장하는 n 변수, 입력의 유효성을 판단하는 check 변수, 홀수 합의 결과를 저장할 odd sum 변수를 선언한다. 처음 n을 입력 받고 isInteger을 통해 적합성을 검사한다. 위의 isInteger 함수 설명에서 보았듯이 함수 내부에서 정상적인 입력인 경우에만 0을 반환하기 때문에 if(isInteger(check, &n) == 0)을 통해 적합성을 검사하고 재입력까지의 과정을 수행한다. 이후 recur\_sum(n)을 통해 홀수 합의 결과를 odd\_sum에 저장하고 결과를 출력한다. 이후 다시 정수 n을 입력 받는다.

이 전체적인 흐름의 결과를 보도록 하자.



1번 입력 및 출력은 첫 문자로 a가 들어온 경우, isInteger에서 검사할 때 양의 정수가 아니기 때문에 오류 출력 및 재입력을 받는다. 이때 -3 음의 정수를 입력했기 때문에 또 다시 오류 출력하고 재입력을 받는다. 올바른 재입력인 경우 그에 해당하는 홀수 합의 결과를 출력한다.

2번 입력 및 출력은 정상적인 경우의 흐름이다.

3번의 경우 첫 입력이 문자, 두 번째 입력도 문자일 경우 오류 출력 및 재입력 받고, 세 번째, 네 번째에는 음의 정수를 입력 받은 경우 오류 출력 및 재입력 받는 것을 확인할 수 있다. 그리고 올바른 입력의 경우 홀수 합의 결과를 출력하는 것을 확인할 수 있다.