< 진수변환 알고리즘 과제 >

이름 : 한아림
학과 : 정보보호학과
학번 : 2022111354
과목 : 컴퓨터 알고리즘
분반 : 01(월 3~4 교시)

< 목차 >

1. 32비트 정수의 표현 범위와 그 이유

이론적으로 32비트 데이터를 처리하는 컴퓨터에서 표현 가능한 정수의 범위는 0 ~ 4,294,967,295 또는 -2,147,483,648 ~ 2,147,483,647 까지이다. 전자의 범위는 0과 자연수만을 포함한 범위이고, 후자는 음수까지 포함한 범위이다.

0 ~ 4,294,967,295의 범위의 경우 4바이트 전체를 숫자 저장 공간으로 이용하기 때문에 2의 32승인 4,294,967,296개의 숫자 중 하나를 저장할 수 있다. 이는 음수를 제외한 32비트 이진수 중 가장 작은 이진수와 가장 큰 이진수 사이의 값이라고 생각할 수 있는데, 음수를 제외한 32비트 이진수 중 가장 작은 값과 가장 큰 값은 아래 표와 같다.

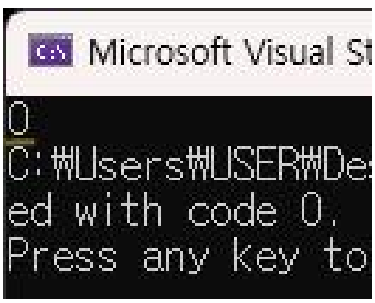
작성상 편의를 위해 지금부터 이진수 00000000000000000000000000000000을 이진수 Min으로, 이진수 11111111111111111111111111111111을 이진수 Max로 표기하겠다.

가장 작은 값	00000000000000000000000000000000
가장 큰 값	11111111111111111111111111111111

이진수 Min과 Max를 각각 10진수로 변환해보면, 결과는 아래 사진과 같다.

```c
#include <stdio.h>
#include <math.h>

int main()
{
    unsigned int m = 0;

    for (int k = 0; k < 32; k++)
    {
        m = pow(2, k)*0 + m;
    }

    printf("%u", m);
```

```c
#include <stdio.h>
#include <math.h>

int main()
{
    unsigned int m = 0;

    for (int k = 0; k < 32; k++)
    {
        m = pow(2, k) + m;
    }

    printf("%u", m);
```



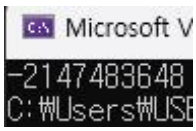
따라서 음수를 제외했을 때 32비트 데이터를 처리하는 컴퓨터에서 표현 가능한 정수 범위는

0 ~ 4,294,967,295 이다.

이번에는 음수를 포함하는 범위인 -2,147,483,648 ~ 2,147,483,647의 경우도 살펴보겠다. 이 범위의 경우 4바이트의 크기이지만 부호가 있는 데이터를 처리하기에, 첫 1비트만 부호로 사용하고 나머지 31비트에는 숫자를 저장한다. 따라서 부호 비트가 0일 경우에는 양수인 2,147,483,648개의 숫자(0 ~ 2,147,483,647) 중 하나를 저장할 수 있다. 반대로 부호 비트가 1일 경우 음수인 2,147,483,648개의 숫자(-2,147,483,648 ~ -1) 중 하나를 저장할 수 있다.

```c
#include <stdio.h>
#include <math.h>

int main()
{
    int m = 0;

    for (int k = 0; k < 31; k++)
    {
        m = pow(2, k) + m;
    }

    printf("%d", m);
}
```


```c
#include <stdio.h>
#include <math.h>

int main()
{
    int m = 0;

    for (int k = 0; k < 31; k++)
    {
        m = pow(2, k) + m;
    }

    printf("%d", -m-1);
}
```



2. 진수 변환 문제 정의

32비트 데이터를 처리하는 컴퓨터에서 표현 가능한 정수의 범위는 0 ~ 4,294,967,295 또는 -2,147,483,648 ~ 2,147,483,647이다. 이 두 범위 중 음수를 포함하는 범위인 -2,147,483,648 ~ 2,147,483,647 안의 10진수 정수 중 하나를 사용자로부터 입력받아 2진수, 8진수, 16진수로 변형하여 모두 출력하는 순서도 및 프로그램을 구현해야 하는 문제이다.

음수의 경우 2의 보수 방법으로 표현해야 한다는 조건이 있으며, 프로그램 구현 언어는 C이다. 프로그램에 꼭 포함되어야 할 경우에 따른 필수 구성은 다음과 같다.

사용자로부터 양수 또는 음수를 입력받는 코드
입력받은 수가 양수일 경우 2진수로의 변환을 구현하는 코드
입력받은 수가 음수일 경우 2진수로의 변환을 구현하는 코드
입력받은 수가 양수일 경우 16진수로의 변환을 구현하는 코드
입력받은 수가 음수일 경우 16진로의 변환을 구현하는 코드
입력받은 수가 양수일 경우 8진수로의 변환을 구현하는 코드
입력받은 수가 음수일 경우 8진수로의 변환을 구현하는 코드
변환한 2, 8, 16진수를 각각 출력하는 코드

추가적으로 진수 간의 변환 과정에 있어 필요한 과정을 구현하는 코드 목록은 아래와 같다.

1의 보수를 구하는 코드
2의 보수를 구하는 코드
수의 절댓값을 구하는 코드

먼저 각 필수 구성 과정의 세부 순서를 살펴보겠다.

< 양수 10진수를 2진수로 변환하는 과정 >
10진수를 2진수로 변환

< 음수 10진수를 2진수로 변환하는 과정 >
입력받은 음수의 절댓값 구하기 -> 절댓값을 2진수로 변환 -> 비트 반전 후 출력

< 양수 10진수를 16진수로 변환하는 과정 >
10진수를 16진수로 변환

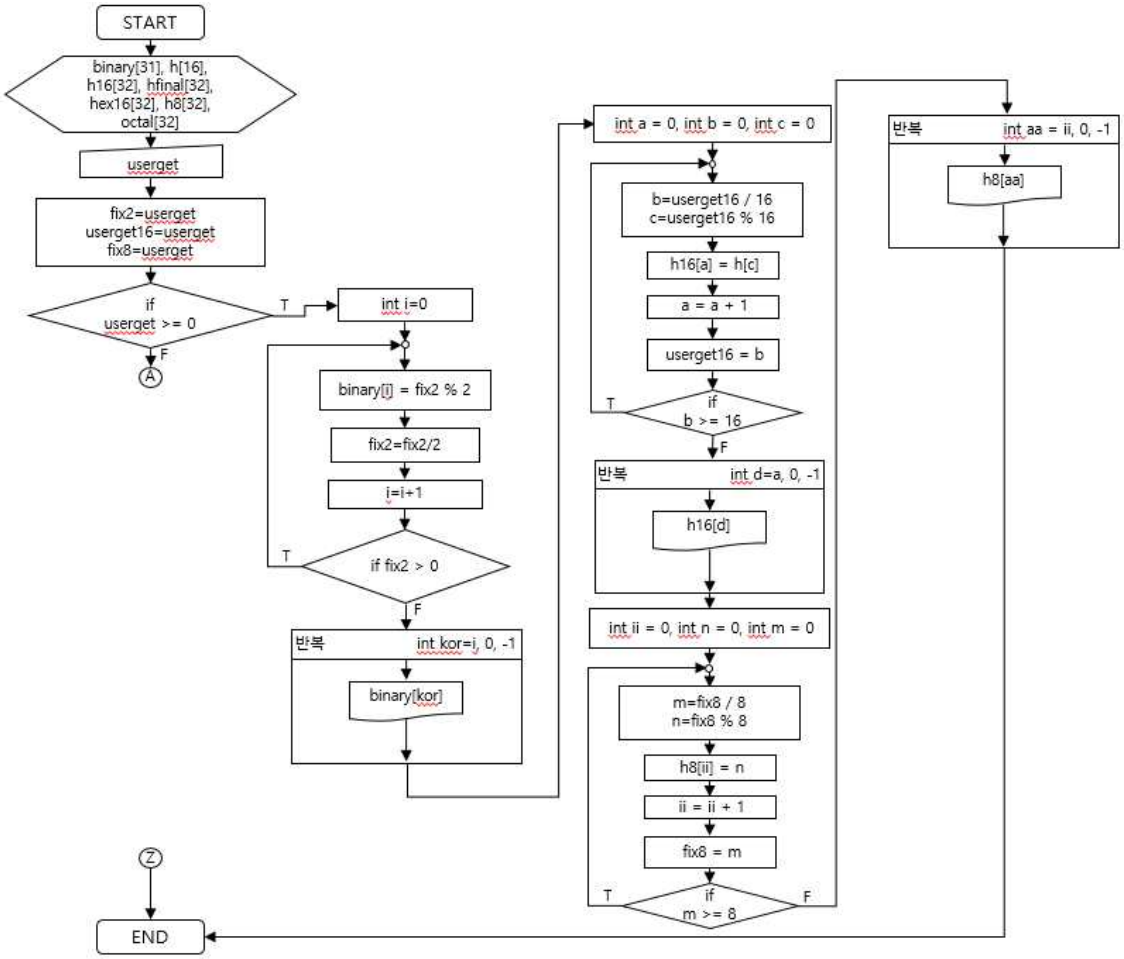
< 음수 10진수를 16진수로 변환하는 과정 >
음수의 절댓값 구하기 -> 절댓값을 2진수로 변환 -> 변환한 2진수에 대한 1의 보수 구하기 -> 변환한 2진수에 대한 2의 보수 구하기 -> 구한 2의 보수를 다시 10진수로 변환 -> 10진수를 16진수로 최종 변환

< 양수 10진수를 8진수로 변환하는 과정 >
10진수를 8진수로 변환

< 음수 10진수를 8진수로 변환하는 과정 >
음수의 절댓값 구하기 -> 절댓값을 2진수로 변환 -> 2진수에 대한 1의 보수 구하기 -> 2진수에 대한 2의 보수 구하기 -> 구한 2의 보수를 10진수로 변환 -> 10진수를 최종적으로 8진수로 변환

실제 위 과정들을 보면 상당수의 과정(1의 보수나 2의 보수 구하기 등)의 중복이 발생함을 확인할 수 있는데, 코드의 통일성을 위하여 동일한 과정이 반복되는 경우는 모두 같은 알고리즘을 사용하였다.

3. 순서도 설계와 설명



순서도는 위의 그림 하나와 아래의 사진 2개, 즉 3개로 이루여져 있다. 편의상 비교적 간단한 양수 변환의 경우를 먼저 설명하겠다.

순서도의 시작은 시작 기호를 통해 선언해 주었다. 그 다음으로는 프로그램 구현에 있어 필요한 배열들을 선언하였다. 선언된 각 배열에 대한 설명은 아래와 같다.

binary[31]	2진수로 변환된 값이 저장될 배열
h[16]	16진수에서 나타날 수 있는 수들을 문자형으로 저장하고 있는 배열
h16[32]	16진수로 변환된 값이 저장될 배열
hfinal[32]	최종적으로 변환된 16진수 값이 저장될 배열로, 사용자로부터 입력받은 값이 음수일 경우에만 사용
hex16[32]	1의 보수와 2의 보수를 구하는 과정에서 쓰일 배열
h8[32]	받아들인 10진수 음수를 8진수로 바꾸는 과정에서 사용될 배열
octal[32]	8진수로 변환된 값이 저장될 배열로, 받아들인 수가 음수일 경우에만 사용

그 다음으로는 키보드 입력 기호를 활용하여 userget 변수에 사용자로부터 값을 입력받는다. 사용자로부터 입력받은 최초의 값인 userget의 값을 보존하기 위해 userget과 동일한 값을 지닌 변수 fix2, fix8, userget16을 정의해주었다. fix2 변수는 10진수와 2진수 사이의 변환 과정에서, fix8은 10진수와 8진수 사이의 변환 과정에서, userget16은 10진수와 16진수 사이의 변환 과정에서 사용된다.

사용자로부터 값을 입력받은 이후에는 조건 기호를 통해 그 값의 부호를 판단한다. 만일 입력받은 값이 0 이상의 양수라면, 곧바로 2, 16, 8진수로의 변환을 하는 과정으로 넘어간다. 반면 입력받은 값이 음수일 경우에는 A로 이동하여 음수 변환을 위한 과정을 수행한다.

입력받은 값이 양수라고 가정했을 때, 먼저 이진수 출력에 쓰일 변수 i를 초기값을 0으로 하여 정의해준다. 그 다음은 입력받은 10진수 양수를 2진수로 변환하는 과정이 실행된다. 미리 선언해둔 배열 binary에 fix2 변수값을 2로 나눈 나머지를 저장한다. 이 과정 후에는 fix2 변수값을 1/2로 줄이며, 변수 i의 값은 1 증가시켜준다. 위 과정을 fix2 의 값이 0 초과일 때 까지만 실행한다. 조건 기호를 통해 fix2의 값이 0 이하가 되었음을 확인했다면, 변환한 2진수를 출력하는 과정으로 넘어간다.

이진수 출력을 위해 반복 변수 kor의 초기값을 i로 하여 선언해준다. 그 후 binary[kor]의 값을 반복해서 출력하며, 한 값의 출력이 끝날 때마다 변수 kor의 값을 -1씩 감소시킨다. 변수 kor이 0이 될 때까지 위 과정을 반복하며, 이진수의 출력이 끝나면 다음 과정으로 넘어간다.

다음은 입력받은 양수 10진수를 16진수로 변환하는 과정이다. 먼저 진수 변환 과정에서 사용할 변수 a, b, c를 각각 초기값을 0으로 하여 정의해준다. 그 이후 변수 b에는 userget16값을 16으로 나눈 몫을 저장하고, 변수 c에는 usesrget16의 값을 16으로 나눈 나머지 값을 저장한다. 그 후 h16[a]에 나머지 c가 나타내고 있는 16진수 문자를 저장하고, a의 값은 1 증가시키고, userget16의 값을 새로운 값인 b로 지정한다. 이후 b의 값이 16 미만이 될 때까지 위 과정을 반복한다.

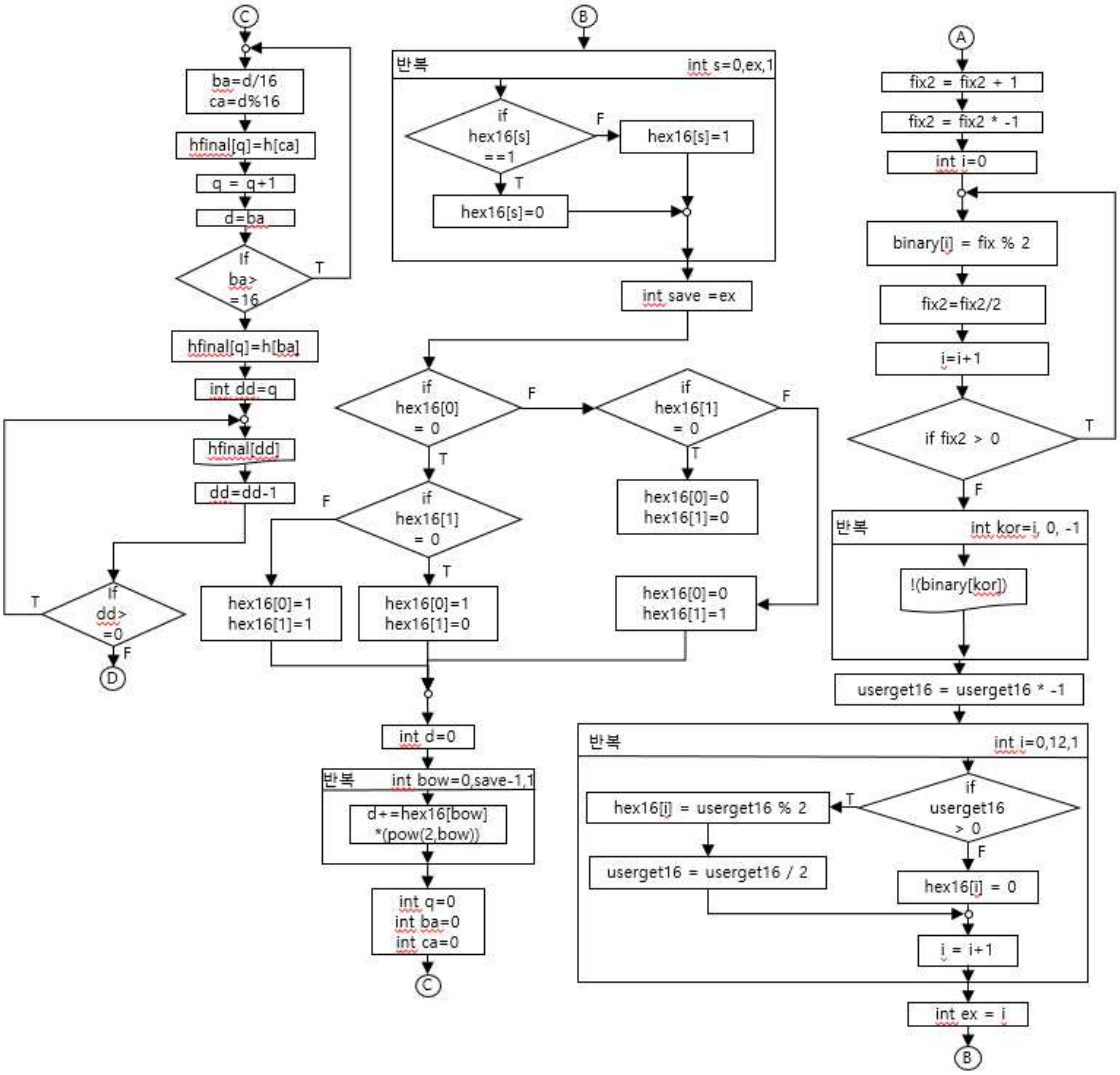
위 과정의 반복이 끝났다면 변환한 16진수를 출력한다. 진수 출력 알고리즘은 저번 2진수 출력 알고리즘과 동일하다. 2진수 출력 알고리즘에서의 변수 kor이 16진수 알고리즘에서는 d, 배열 binary[]이 16진수 알고리즘에서는 h16으로 변경되었다. h16[d]의 값을 d가 0이 될 때까지 출력했다면, 다음 과정인 8진수 변환으로 이동한다.

8진수 변환도 마찬가지로 진수 변환 과정에서 쓰일 변수인 ii, n, m의 초기값을 0으로 하여 각각 선언해준다. 그 후 입력받은 양수 10진수를 8진수로 변환하는 과정을 거친다. 변수 m에 fix8을 8로 나눈 몫을, 변수 n에 fix8을 8로 나눈 나머지 값을 보관한다. 그 후 h8[ii]에 n의 값을 보관하고, ii의 값을 1 증가시킨다. 그 후 fix8의 값을 새로운 값인 m으로 바꾸어주고, 위의 과정들을 m이 8미만의 값이 될 때까지 반복한다.

위 과정이 끝났다면 변환이 완료된 8진수를 출력한다. 8진수를 출력하는 알고리즘은 2진수

출력 알고리즘과 16진수 출력 알고리즘과 동일하다. h8[aa] 값의 출력을 반복 변수 aa의 값이 0이 될 때까지 반복하고, 한 값의 출력이 끝날 때마다 aa의 값을 1씩 감소시킨다.

위의 모든 과정이 끝났다면, 완료 기호로 순서도를 끝마친다.



만일 앞선 경우와 달리 입력받은 10진수가 음수라면, A로 이동하여 아래의 과정을 수행한다. 입력받은 10진수 음수에 1을 더한 뒤 그 절댓값을 구하고, 이진수 출력을 위한 변수 i를 정의해준다.

그 다음 구한 절댓값을 2진수로 변환하는 과정을 진행한다. binary[i]에 fix2를 2로 나눈 나머지를 저장하고, fix2의 값을 절반으로 줄인다. 그 후 i의 값을 1 증가시키고, 이를 fix2의 값이 0 이하가 될 때까지 반복한다.

위 과정이 끝나면 반복변수 kor을 초기값이 i가 되도록 지정하고, binary[kor]의 값을 ! 연산자를 통해 비트가 반전된 상태로 출력해준다. 값이 하나 출력될때마다 kor의 값을 1씩 감소시키고, kor의 값이 0이 될 때까지 위 과정을 반복한다.

다음은 16진수로 변환 후 출력하는 과정이다. 16진수 변환의 경우에도 입력받은 음수의 절댓값을 구하고, 반복 변수 i의 초기값이 0이 되도록 지정한다. 만일 userget16이 0 초과의

값을 가진다면 hex16[i]에 userget16을 2로 나눈 나머지를 저장하고, userget16의 값을 1/2로 감소시킨다. 이후 변수 i의 값을 1 증가시킨다. 만일 userget16의 값이 0이하의 값을 가진다면 hex16[i]에 0을 지정하고 I를 1 증가시킨다. 위의 과정을 i의 값이 12가 될 때까지 반복한다.

위 과정의 반복 이후 변수 ex의 초기값을 i로 하여 정의한 후 B로 이동한다. B로 이동한 후 hex16 배열에 대한 1의 보수를 구하는 과정을 수행한다. 먼저 반복 변수 s의 초기값이 0이 되도록 설정한다. 이후 hex16[s]가 1일 경우에는 hex16[s]에 0을 지정하고, 1이 아닐 경우에는 hex16[s]에 1을 지정한다. 이 과정을 s가 ex-1이 될 때까지 반복하며, 한 과정이 끝날 때마다 s의 값을 1씩 증가시킨다.

위의 과정이 끝났다면, 새로운 변수 save를 선언하고 이 변수에 ex의 값을 저장한다. 이후 hex16배열에 대한 2의 보수를 구하는 과정이 수행된다. hex16[0], hex16[1]이 모두 0인 경우, hex16[0]이 0이고 hex16[1]이 1인 경우, hex16[0]이 1이고 hex16[1]이 0인 경우, hex16[0]과 hex16[1]이 모두 1인 경우로 나누어 각 경우에 맞는 값을 지정해준다.

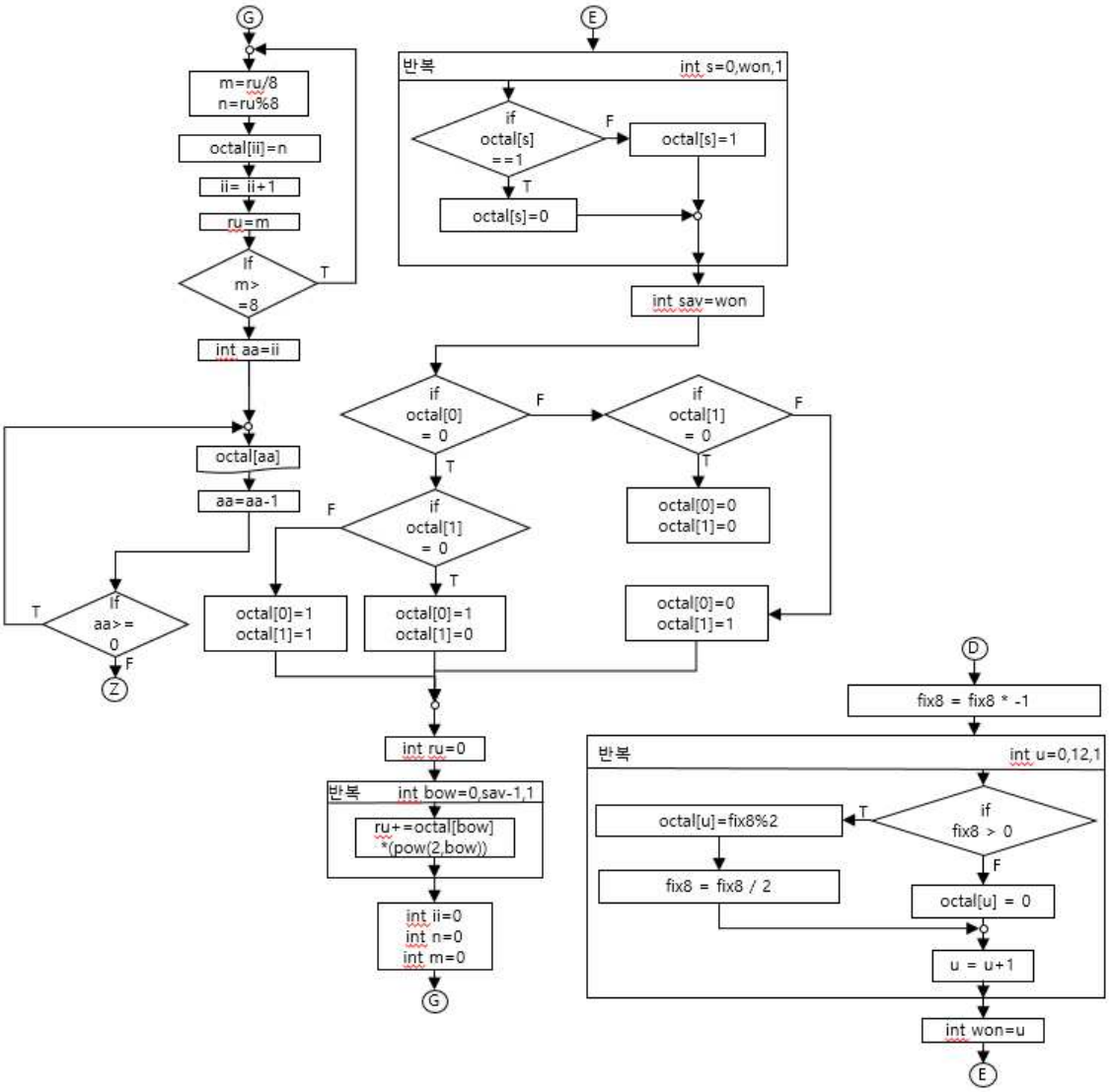
hex16[0], hex16[1]이 모두 0인 경우, hex16[0]과 hex16[1]에 각각 1, 0을 지정해준다.
hex16[0]이 0이고 hex16[1]이 1인 경우, hex16[0]과 hex16[1]에 각각 1, 1을 지정해준다.
hex16[0]이 1이고 hex16[1]이 0인 경우, hex16[0]과 hex16[1]에 각각 0, 0을 지정해준다.
hex16[0]과 hex16[1]이 모두 1인 경우, hex16[0]과 hex16[1]에 각각 0, 1을 지정해준다.

이후 새로운 변수 d를 초기값을 0으로 하여 지정해주고 이전 과정에서 구한 2의 보수를 10진수로 변환하는 과정을 수행한다. 반복 변수 bow의 초기값을 0으로 하고, bow가 save-1의 값이 될 때까지 d에 d와 hex16[bow]와 2의 bow승을 곱한 값을 더해준다. 한 과정이 끝나면 bow의 값을 1씩 더한다.

그 후 10진수 d를 16진수로 변환하기 위해 변수 ba, q, ca의 초기값을 각각 0으로 하여 선언하고, C로 이동한다.

C로 이동한 뒤 ba에 d를 16으로 나눈 몫의 값을 저장하고, ca에는 d를 16으로 나눈 나머지를 저장한다. 이후 hfinal[q]에 h[ca]의 값을 저장하고, q의 값을 1 증가시키고 d에 ba의 값을 할당한다. 이 과정을 ba의 값이 16미만이 될 때까지 반복한다.

이후 hfinal[q]에 h[ba]의 값을 저장하고, 새로운 변수 dd에 q의 값을 저장한다. 그 다음 최종적으로 변환한 16진수를 출력하는 과정을 수행한다. hfinal[dd]를 한번 출력할 때마다 dd의 값을 1 감소시키고, dd의 값이 음수가 될 때까지 반복한다. 반복이 끝나면 D로 이동한다.



D로 이동한 뒤 음수 10진수를 8진수로 변환하는 과정을 수행한다. 먼저 입력받은 음수의 절댓값을 구하고, 반복 변수 u의 초기값을 0으로 하여 반복 과정을 수행한다. 만일 fix8의 값이 0 초과일 경우 octal[u]에 fix8을 2로 나눈 나머지 값을 저장하고, fix8의 값을 1/2로 감소시킨다. 이후 u의 값을 1 증가시킨다. 만일 fix8의 값이 0 이하라면 octal[u]에 0을 저장하고, 마찬가지로 u의 값을 1 증가시킨다. 위 과정을 u가 12가 될 때까지 반복한다.

위 과정의 반복이 끝나면 변수 won의 초기값을 u로 설정하여 정의하고, E로 이동한다. E로 이동한 후 octal에 대한 1의 보수를 구하는 과정이 진행된다. 반복변수 s의 초기값을 0으로 하고, 만일 octal[s]의 값이 1이라면 octal[s]에 0을 저장하고 s를 1 증가시킨다. 만일 octal[s]의 값이 1이 아니라면 octal[s]에 1을 저정하고 s를 1 증가시킨다. 이 과정을 s가 won이 될 때까지 반복한다.

이후 변수 sav의 초기값을 won으로 하여 정의하고, octal에 대한 2의 보수를 구하는

과정으로 이동한다. 2의 보수 알고리즘은 16진수의 경우와 완벽히 같으나, 8진수 경우에 대해서는 배열 hex16 대신 octal을 사용한다.

2의 보수를 구하는 과정이 끝나면 새로운 변수 ru의 초기값이 0이 되도록 설정해주고, 2의 보수를 10진수로 변환하는 과정을 수행한다. 반복 변수 bow가 0부터 sav-1이 될 때까지 1씩 증가시키며 ru에 octal[bow]와 2의 bow승을 곱한 값을 누적하는 과정을 수행한다.

위의 과정이 끝나면 변환된 10진수를 다시 8진수로 변환하는 작업을 실행한다. 새로운 변수 ii와 n,m 의 초기값을 각각 0이 되도록 설정한다. 그 후 m에 ru를 8로 나눈 몫을, n에 ru를 8로 나눈 나머지를 저장하고, octal[ii]에 n을 저장한다. 이후 ii의 값을 1 증가시키고, ru에 m을 저장한다. 이 과정들을 m이 8미만이 될 때까지 실행한다.

이후 octal[ii]에 m을 저장하고, 변수 aa를 선언하고 ii의 값을 저장한다. 마지막으로 변환한 8진수를 출력하는 과정을 수행한다. octal[aa]를 한번 출력할 때마다 aa의 값을 1씩 감소시키고, aa가 0 미만이 될 때까지 위 과정을 반복한다.

위의 모든 과정이 끝나면 Z로 이동하고, 순서도는 끝나게 된다.

5. 순서도 구현과 프로그램 설명

```c
#include <stdio.h>
#include <math.h>
#include <limits.h>

int binary[31];         //2진수로 변환된 값이 저장될 배열
int userget = 0;        //사용자로부터 입력받는 정수
char h[16] = { '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F' };   //16진수에서 나타날 수 있는 숫자를 문자형으로 저장하고 있는 배열
char h16[32];  //16진수로 변환된 값이 저장될 배열
char hfinal[32];    //최종적으로 변환된 16진수 값이 저장될 배열 -> 사용자로부터 입력받은 값이 음수일 경우에만 사용
int hex16[32];    //1의 보수와 2의 보수를 구하는 과정에서 쓰일 배열
int h8[32];     //받아들인 10진수 양수 --> 8진수로 변환된 값이 저장될 배열
int octal[32];  //받아들인 10진수 음수 --> 8진수 과정에서만 사용
                //8진수로 변환된 값이 저장될 배열


int main()
{
    //사용자로부터 정수 입력받기
    printf("정수를 입력하시오 : ");
    scanf_s("%d", &userget);

    //원래 입력값 보존을 위한 별도 변수 정의
    int fix2 = userget;       //10 -> 2진수 때 사용
    int userget16 = userget;  //10 -> 16진수 때 사용
    int fix8 = userget;       //10 -> 8진수 때 사용

    //입력받은 양수 10진수 --> 2진수
    if (fix2 >= 0)
    {
        int i = 0;     //이진수 출력을 위한 변수 i

        while (fix2 > 0)
        {
            binary[i] = fix2 % 2;  //배열 binary에 fix2를 2로 나눈 나머지 저장
            fix2 = fix2 / 2;    //fix2를 2로 나눈 몫이 새로운 fix2가 됨
            i++;
        }
```

```c
        //변환한 2진수값 출력
        printf("2진수 : ");
        for (int kor=i; kor >= 0; kor--)
        {
            printf("%d", binary[kor]);
        }

    }

    //입력받은 음수 10진수 --> 2진수
    else
    {
        int i = 0;       //이진수 출력을 위한 변수 i

        fix2 = fix2 + 1;
        fix2 = fix2 * -1;   //입력받은 음수의 절댓값 구하기

        while (fix2 > 0)    //절댓값을 이진수로 변환
        {
            binary[i] = fix2 % 2;
            fix2 = fix2 / 2;
            i++;
        }
        int kor = i;

        printf("2진수 : ");
        for (kor = i; kor >= 0; kor--)
        {
            printf("%d", !(binary[kor]));    //! 연산자를 사용하여 비트 반전시켜 출력
        }

    }
```

```
71
72      printf("\n");       //줄 바꿈
73
74      //입력받은 음수 10진수 --> 16진수
75      if (userget16 < 0)
76      {
77          userget16 = userget16 * -1;  //입력받은 10진수 음수의 절댓값 구하기
78          int i = 0;
79
80          while (i<=12)    //절댓값을 이진수로 변환
81          {
82              if (userget16 > 0)
83              {
84                  hex16[i] = userget16 % 2;
85                  userget16 = userget16 / 2;
86                  i++;
87              }
88              else
89              {
90                  hex16[i] = 0;
91                  i++;
92              }
93
94          }
95          int ex = i;
96
97          for (int s =0; s <ex; s++)      //hex16[] 에 대한 1의 보수 구하기
98          {
99              if (hex16[s] == 1)
100             {
101                 hex16[s] = 0;
102             }
103             else
104             {
105                 hex16[s] = 1;
106             }
107         }
108
109         int save = ex;
110
111             if (hex16[0] == 0 && hex16[1] == 0)        //hex16[]에 대한 2의 보수 구하기(이진수 덧셈 => 십진수 1, 즉 이진수 01 을 더하기)
112             {
113                 hex16[0] = 1; hex16[1] = 0;
114             }
115             else if (hex16[0] == 0 && hex16[1] == 1)
116             {
117                 hex16[0] = 1; hex16[1] = 1;
118             }
119             else if (hex16[0] == 1 && hex16[1] == 0)
120             {
121                 hex16[0] = 0; hex16[1] = 0;
122             }
123             else
124             {
125                 hex16[0] = 0; hex16[1] = 1;
126             }
127
128             int d = 0;
129             for (int bow = 0; bow < save - 1; bow++)   //2의 보수를 10진수로 변환
130             {
131                 d = d + hex16[bow] * (pow(2, bow));
132             }   //d는 변환된 10진수
133
134             int q = 0; int ba; int ca;      //10진수 d를 16진수로 변환
135             do
136             {
137                 ba = d / 16;
138                 ca = d % 16;
139                 hfinal[q] = h[ca];     //최종적으로 변환된 16진수는 hfinal[]에 저장됨
140                 q++;
141                 d = ba;
142             } while (ba >= 16);
143
144             hfinal[q] = h[ba];
145             int dd = q;
```

```
        //최종적으로 변환한 16진수 출력
        printf("16진수 : 0x");
        do
        {
            printf("%c", hfinal[dd]);
            dd--;
        } while (dd >= 0);

    }

    //입력받은 양수 10진수 --> 16진수
    else
    {
        int a = 0; int b; int c;
        do
        {
            b = userget16 / 16;
            c = userget16 % 16;
            h16[a] = h[c];
            a++;
            userget16 = b;
        } while (b >= 16);

        h16[a] = h[b];
        int d = a;

        //변환한 16진수값 출력
        printf("16진수 : ");
        do
        {
            printf("%c", h16[d]);
            d--;
        } while (d >= 0);

    }

    printf("\n");
```

```
    //입력받은 음수 10진수 --> 8진수
    if (fix8 < 0)
    {
        fix8 = fix8 * -1;  //입력받은 10진수 음수의 절댓값 구하기
        int u = 0;

        while (u <= 12)    //절댓값을 이진수로 변환
        {
            if (fix8 > 0)
            {
                octal[u] = fix8 % 2;
                fix8 = fix8 / 2;
                u++;
            }
            else
            {
                octal[u] = 0;
                u++;
            }

        }
        int won = u;

        for (int s = 0; s < won; s++)      //octal[] 에 대한 1의 보수 구하기
        {
            if (octal[s] == 1)
            {
                octal[s] = 0;
            }
            else
            {
                octal[s] = 1;
            }
        }

        int sav = won;
```

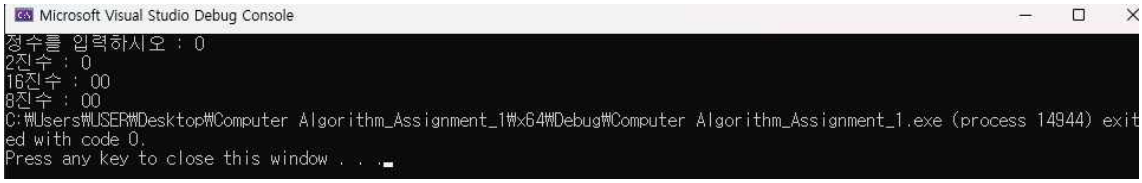
```
        if (octal[0] == 0 && octal[1] == 0)        //octal[]에 대한 2의 보수 구하기(이진수 덧셈 => 십진수 1, 즉 이진수 01 을 더하기)
        {
            octal[0] = 1; octal[1] = 0;
        }
        else if (octal[0] == 0 && octal[1] == 1)
        {
            octal[0] = 1; octal[1] = 1;
        }
        else if (octal[0] == 1 && octal[1] == 0)
        {
            octal[0] = 0; octal[1] = 0;
        }
        else
        {
            octal[0] = 0; octal[1] = 1;
        }

        int ru = 0;
        for (int bow = 0; bow < sav - 1; bow++)   //2의 보수를 10진수로 변환
        {
            ru = ru + octal[bow] * (pow(2, bow));
        }   //ru는 변환된 10진수

        int ii = 0; int n; int m;       //변환된 10진수를 최종적으로 8진수로 변환
        do
        {
            m = ru / 8;
            n = ru % 8;
            octal[ii] = n;
            ii++;
            ru = m;
        } while (m >= 8);

        octal[ii] = m;
        int aa = ii;
```

```
        //최종적으로 변환한 8진수 출력
        printf("8진수 : ");
        do
        {
            printf("%d", octal[aa]);
            aa--;
        } while (aa >= 0);
    }

    //입력받은 양수 10진수 --> 8진수
    else
    {
        int ii = 0; int n; int m;
        do
        {
            m = fix8 / 8;
            n = fix8 % 8;
            h8[ii] = n;
            ii++;
            fix8 = m;
        } while (m >= 8);

        h8[ii] = m;
        int aa = ii;

        //최종적으로 변환한 8진수 출력
        printf("8진수 : ");
        do
        {
            printf("%d", h8[aa]);
            aa--;
        } while (aa >= 0);
    }

    return 0;
}
```

5. 실행화면 상세

- 입력받은 수가 양수인 경우

- 입력받은 수가 0인 경우



- 입력받은 수가 음수인 경우