알고리즘 과제 #1

컴퓨터 알고리즘 02분반, 정보보호학과, 2020111340, 우은지

1. 32비트 데이터를 처리하는 컴퓨터에서 표현 가능한 정수의 범위

32비트 데이터를 처리하는 컴퓨터에서 표현 가능한 정수의 범위는 컴퓨터는 수를 이진법을 이용해서 0과 1로 표현하기 때문에 한 비트당 2가지씩 표현할 수 있어 signed int의 경우엔 -2^(32-1) ~ 2^(32-1)-1만큼의 범위인 −2,147,483,648부터 2,147,483,647까지 표현 가능하고 unsigned int의 경우 0부터 4,294,967,295까지 표현 가능하다.

2. 진수변환 문제 정의

main 함수의 문제 정의

1. 사용자에게 −2,147,483,648부터 2,147,483,647 사이의 값을 입력받는다
2. 사용자에게 입력받은 값이 −2,147,483,648부터 2,147,483,647 사이에 있는 값인지 확인한다.
3. −2,147,483,648부터 2,147,483,647 사이에 있는 값이 아니라면 위의 과정을 반복하고 사이에 있는 값이라면 다음 단계로 넘어간다.
4. 사용자에게 입력받은 값을 인자로 넣어 2진수 구해서 출력하고 반환해주는 서브루틴을 거치고 온다.
5. 4번에서 구한 값을 인자로 넣어 8진수 구해서 출력해주는 서브 루틴을 거치고 온다.
6. 4번에서 구한 값을 인자로 넣어1 6진수 구해서 출력해주는 서브 루틴을 거치고 온다.

2진수 구해서 출력해주는 서브루틴의 문제 정의

1. 인자로 받은 사용자의 입력값이 음수인지 양수인지 판단한다.
2. 양수라면 몫이 0이 될 때까지 2로 나눠주고 나누면서 생긴 나머지들을 배열에 저장한다.
3. 음수라면 양수와 같이 계산한 다음 배열에 저장된 값을 1의 보수를 구하고 1을 더해 2의 보수법으로 표현해 그 값을 저장한다.
4. 구한 값을 출력한다.

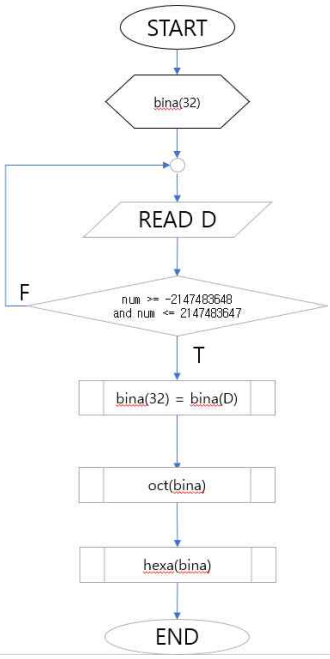
8진수 구해서 출력해주는 서브루틴의 문제정의

1. 2진수 값을 뒤에서 3자리씩 끊어 가중치를 곱해 더한 후 그 값을 배열에 저장한다.
2. 구한 8진수를 출력한다.

16진수 구해서 출력해주는 서브루틴의 문제정의

1. 2진수 값을 뒤에서 4자리씩 끊어 가중치를 곱해 더한 후 그 값을 배열에 저장한다.
2. 구한 16진수를 출력한다.

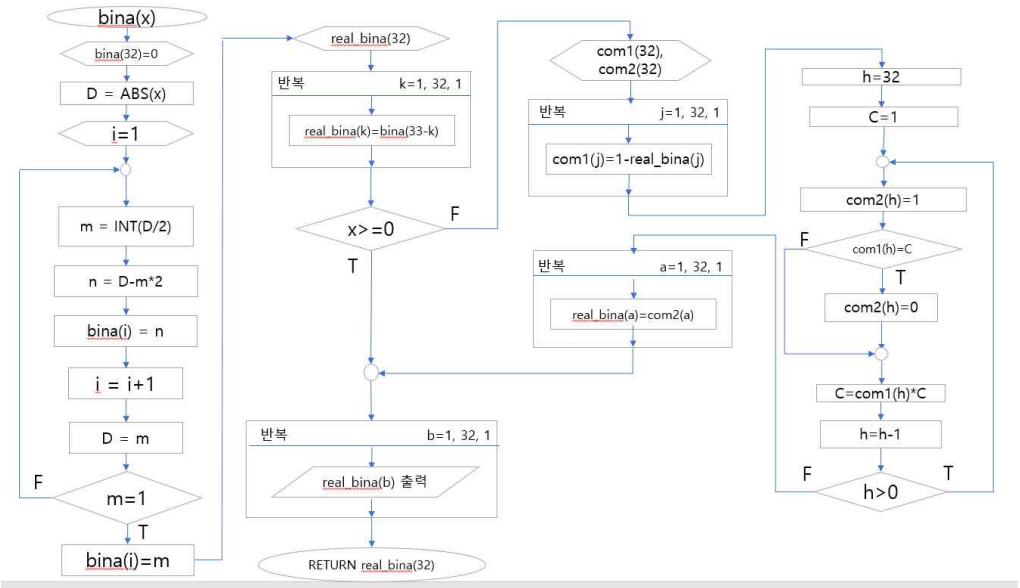
3. 순서도와 설명



2진수로 변환 후 그 값을 이용해 서브루틴을 호출해 8진수, 16진수로의 값을 구할 것이기 때문에 2진수를 변환한 값을 저장할 bina(32)라는 배열을 선언해 준다.

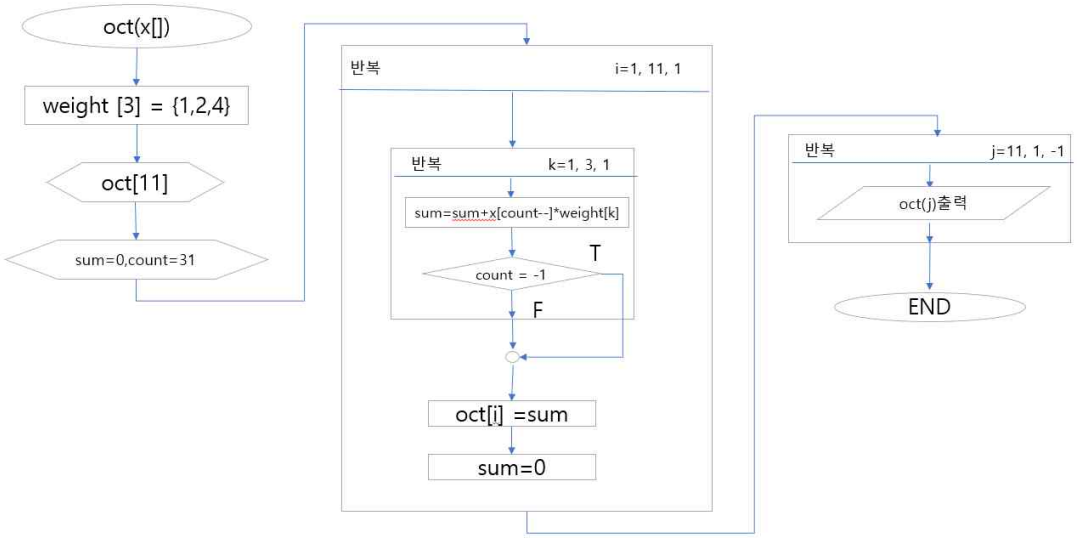
사용자에게 값을 입력받는다. 32비트 컴퓨터로 표현 가능한 범위를 벗어나는 값을 입력받았을 경우 다시 입력받는다.

bina라는 서브루틴을 통해 2진수로 표현된 값을 구하고 출력한다.

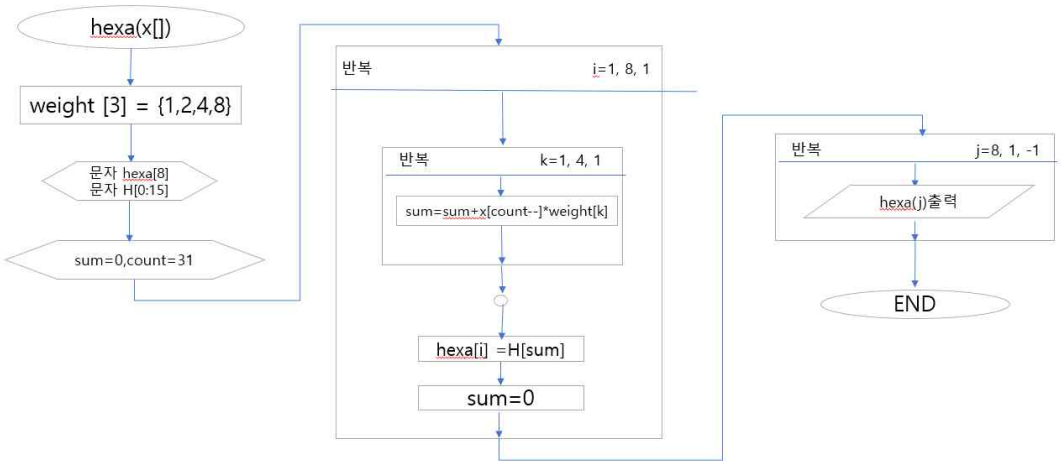


2진수로 표현한 값을 저장할 bina라는 배열을 선언하고
매개변수로 받은 사용자 입력 값을 절댓값을 씌운 값을 D에 저장한다.

2로 나누며 나머지를 배열에 저장하여 입력값의 절댓값을 2진수로 표현한다.
2진수의 값이 거꾸로 저장되었기 때문에 제대로 된 순서로 real_bina 배열에 저장한다.
절댓값을 씌우기 전 매개변수로 받은 값이 양수일 경우 real_bina를 바로 출력한다.
절댓값을 씌우기 전 값이 음수일 경우 1의 보수를 구해 com1 배열에 저장 후
1의 보수를 이용해 2의 보수를 구해 com2에 저장한다.
com2를 real_bina에 복사하고 real_bina를 출력하고 마친다.



다음은 8진수로 변환하는 서브루틴이다. 매개변수로는 사용자에게 입력받은 값을 2진수로 변환하여 저장한 배열을 받는다.
3자리씩 끊어서 가중치를 곱한 다음 더해서 2진수를 8진수로 변경한 후 oct 배열을 출력하고 마친다.
(32비트로 표현 가능한 수를 8진수로 바꾸려면 int(32/3)+1 하면 11번이기 때문에 반복문을 11번 돌면서 구한다.)



16진수로 변환해주는 서브루틴이다. 8진수로 변환해주는 서브루틴과 동일하게 매개변수를 받아온다.
16진수의 값이 0부터 F까지 저장된 배열 H를 선언해주고 16진수로 변환한 값을 저장할 hexa 배열을 선언한다.

가중치를 곱해 더하면서 값을 구하고, 해당 합을 배열 H를 이용해 16진수로 변환한 후 출력하고 서브루틴을 마친다.

4. 구현(프로그램 소스 C)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int* bina(int x)  // 입력받은 정수를 2진수로 바꿔 출력하고 return 해주는 함수
{
    int *bina = (int*)malloc(sizeof(int)*32);
    memset(bina, 0, sizeof(int) * 32);

    int D = abs(x);
    int i = 0;

    int m, n;

    while (1) {
        m = int(D / 2);
        n = D - m * 2;
        bina[i] = n;  // 2로 값을 나누면서 나머지를 2진수를 저장할 배열에 저장
        i = i + 1;
        D = m;

        if (m == 1 or m==0) { // 나눌 수 있을 때까지 나눠서 m이 1이 되거나 입력 값이 1이
여서 처음 m이 0인 경우
            bina[i] = m;
            break;
        }
    }

    int* real_bina = (int*)malloc(sizeof(int) * 32);

    for (int k = 0; k < 32; k++) { // 역순으로 저장된 2진수를 제대로 바꿔줌
        real_bina[k] = bina[31 - k];
    }

    if (x > 0) {  // 입력값이 양수일 경우
        printf("2진수:");
        for (int b = 0; b < 32; b++)
            printf("%d", real_bina[b]);
    }
    else {   // 입력값이 음수일 경우
        int* com1 = (int*)malloc(sizeof(int) * 32);
```

```
        int* com2 = (int*)malloc(sizeof(int) * 32);

        for (int j = 0; j <32; j++) {  // 1의 보수를 구함
            com1[j] = 1 - real_bina[j];
        }

        int h = 31;
        int C = 1;

        while (1) {
            com2[h] = 1;

            if (com1[h] == C)
                com2[h] = 0;
            C = com1[h] * C; // 캐리와 해당 위치의 1의 보수 값이 같은 경우에만 캐리가 1이
된

            h = h - 1;

            if(h < 0)
                break;
        }
;
        for (int a = 0; a < 32; a++) { // 마지막에 return을 real_bina로 해주기 위해 2의 보
수 값을 옮겨줌
            real_bina[a] = com2[a];
        }

        printf("2진수:");
        for (int b = 0; b <32; b++)
            printf("%d", com2[b]);

    }
    return real_bina;
}

int oct(int x[]) {  // 입력받은 정수를 8진수로 바꿔 출력해주는 함수
    int weight[3] = {1, 2, 4};
    int oct[11] = {};
    int sum = 0;
    int count = 31;

    for (int i = 0; i < 11; i++) { // 32비트이기 때문에 32비트로 표현 가능한 10진수를 8진수로
```

```
바꾸려면 11자리로 표현 가능
        for (int k = 0; k < 3; k++) {
            sum = sum + x[count--] * weight[k];  // 가중치를 곱해 누적해서 더함(8진수는
3자리씩이니까 3번씩 반복문을 돈다.)

            if (count == -1) {
                break;
            }
        }

        oct[i] = sum;
        sum = 0;
    }

    printf("\n\n8진수:");
    for (int j = 10; j >= 0; j--) { // 거꾸로 저장된 8진수를 제대로 출력
        printf("%d", oct[j]);
    }

    return 0;
}


int hexa(int x[]) {  // 입력받은 정수를 16진수로 바꿔 출력해주는 함수
    int weight[4] = {1, 2, 4, 8 };
    char hexa[8] = {};
    int sum = 0;
    char H[16] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
    int count = 31;

    for (int i = 0; i < 8; i++) {
        for (int k = 0; k < 4; k++) {
            sum = sum + x[count--]*weight[k];
        }

        hexa[i] = H[sum]; // 16진수로 바꿔서 hexa 배열에 저장
        sum = 0; // 4자리를 읽고 다시 sum을 0으로 초기화
    }

    printf("\n\n16진수:");
    for (int k = 7; k >=0; k--) {
        printf("%c", hexa[k]);
    }
```

```
    return 0;
}

int main(void) {

    int num;

    while (1) {
        printf("정수를 입력하세요: ");
        scanf_s("%d", &num);

        if (num >= -2147483648 and num <= 2147483647) // 절대값을 구한 후 먼저 2진수로 변환하기 때문에 -2147483648이 아닌 -2147483647까지만 표현 가능
            break;
        else
            printf("-2147483648 ~ 2147483647 사이의 정수를 입력해주세요.");

    }

    if (num == -2147483648) {  // 절대값을 구한 후 먼저 2진수로 변환하기 때문에 2진수로 -2147483648이 아닌 -2147483647까지만 표현 가능 따라서 -2147483648는 이렇게 따로 지정해줘야함
        printf("2진수:10000000000000000000000000000000");
        printf("8진수:60000000001");
        printf("16진수:80000000");
    }
    else {
        int* a;
        a = bina(num);

        oct(a);
        hexa(a);
    }

    return 0;
}
```

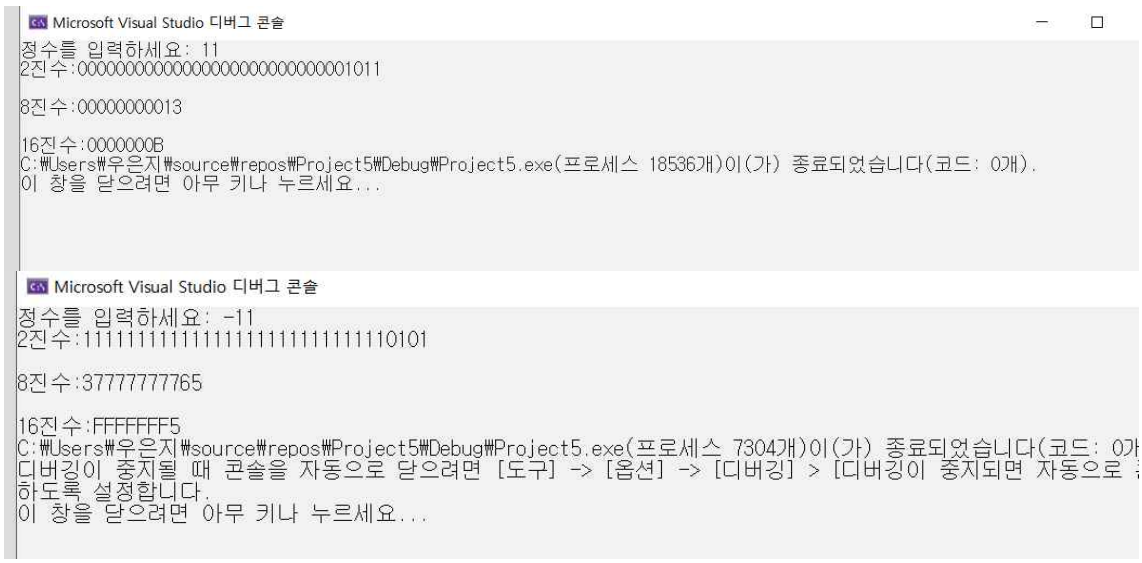
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int* bina(int x)  // 입력받은 정수를 2진수로 바꿔 출력하고 return 해주는 함수
{
    int *bina = (int*)malloc(sizeof(int)*32);
    memset(bina, 0, sizeof(int) * 32);

    int D = abs(x);
    int i = 0;

    int m, n;

    while (1) {
        m = int(D / 2);
        n = D - m * 2;
        bina[i] = n;  // 2로 값을 나누면서 나머지를 2진수를 저장할 배열에 저장
        i = i + 1;
        D = m;

        if (m == 1 or m==0) { // 나눌 수 있을 때까지 나눠서 m이 1이 되거나 입력 값이 1이여서 처음 m이 0인 경우
            bina[i] = m;
            break;
        }
    }

    int* real_bina = (int*)malloc(sizeof(int) * 32);

    for (int k = 0; k < 32; k++) { // 역순으로 저장된 2진수를 제대로 바꿔줌
        real_bina[k] = bina[31 - k];
    }

    if (x > 0) {  // 입력값이 양수일 경우
        printf("2진수: ");
        for (int b = 0; b < 32; b++)
            printf("%d", real_bina[b]);
    }
    else {   // 입력값이 음수일 경우
        int* com1 = (int*)malloc(sizeof(int) + 32);
        int* com2 = (int*)malloc(sizeof(int) + 32);

        for (int j = 0; j <32; j++) {  // 1의 보수를 구함
            com1[j] = 1 - real_bina[j];
        }
            ,

            int h = 31;
            int C = 1;

            while (1) {
                com2[h] = 1;

                if (com1[h] == C)
                    com2[h] = 0;
                C = com1[h] * C; // 캐리와 해당 위치의 1의 보수 값이 같은 경우에만 캐리가 1이됨

                h = h - 1;

                if(h < 0)
                    break;
            }
    ;
            for (int a = 0; a < 32; a++) { // 마지막에 return을 real_bina로 해주기 위해 2의 보수 값을 옮겨줌
                real_bina[a] = com2[a];
            }

            printf("2진수: ");
            for (int b = 0; b <32; b++)
                printf("%d", com2[b]);

        }
        return real_bina;
    }
```

```c
int oct(int x[]) {  // 입력받은 정수를 8진수로 바꿔 출력해주는 함수
    int weight[3] = {1, 2, 4};
    int oct[11] = {};
    int sum = 0;
    int count = 31;

    for (int i = 0; i < 11; i++) { // 32비트이기 때문에 32비트로 표현 가능한 10진수를 8진수로 바꾸려면 11자리로 표현 가능
        for (int k = 0; k < 3; k++) {
            sum = sum + x[count--] * weight[k];  // 가중치를 곱해 누적해서 더함(8진수는 3자리씩이니까 3번씩 반복문을 돈다.)

            if (count == -1) {
                break;
            }
        }

        oct[i] = sum;
        sum = 0;
    }

    printf("₩n₩n8진수:");
    for (int j = 10; j >= 0; j--) { // 거꾸로 저장된 8진수를 제대로 출력
        printf("%d", oct[j]);
    }

    return 0;
}
```

```c
int hexa(int x[]) {  // 입력받은 정수를 16진수로 바꿔 출력해주는 함수
    int weight[4] = {1, 2, 4, 8 };
    char hexa[8] = {};
    int sum = 0;
    char H[16] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
    int count = 31;

    for (int i = 0; i < 8; i++) {
        for (int k = 0; k < 4; k++) {
            sum = sum + x[count--]*weight[k];
        }

        hexa[i] = H[sum]; // 16진수로 바꿔서 hexa 배열에 저장
        sum = 0; // 4자리를 읽고 다시 sum을 0으로 초기화
    }

    printf("₩n₩n16진수:");
    for (int k = 7; k >=0; k--) {
        printf("%c", hexa[k]);
    }

    return 0;
}
```

```c
int main(void) {

    int num;

    while (1) {
        printf("정수를 입력하세요: ");
        scanf_s("%d", &num);

        if (num >= -2147483648 and num <= 2147483647) // 절대값을 구한 후 먼저 2진수로 변환하기 때문에 -21474836480이 아닌 -2147483647까지만 표현 가능
            break;
        else
            printf("-2147483648 ~ 2147483647 사이의 정수를 입력해주세요.");

    }

    if (num == -2147483648) {  // 절대값을 구한 후 먼저 2진수로 변환하기 때문에 2진수로 -21474836480이 아닌 -2147483647까지만 표현 가능 따라서 -2147483648는 이렇게 따로 지정해줘야함
        printf("2진수:10000000000000000000000000000000");
        printf("8진수:60000000001");
        printf("16진수:80000000");
    }
    else {
        int* a;
        a = bina(num);

        oct(a);
        hexa(a);
    }

    return 0;
}
```

5. 실행화면



***카피킬러 유사도 검사**