Infinity-Calculator

IC-PBL 6조

김태현, 류관우, 유훈, 김선엽

1. 제작 환경 (운영체제) :

Ubuntu 18.04

2. 컴파일 방법(inf_calc 이용 가능) : gcc 설치 필수

gcc -o inf_calc *.c

3. 실행 방법 :

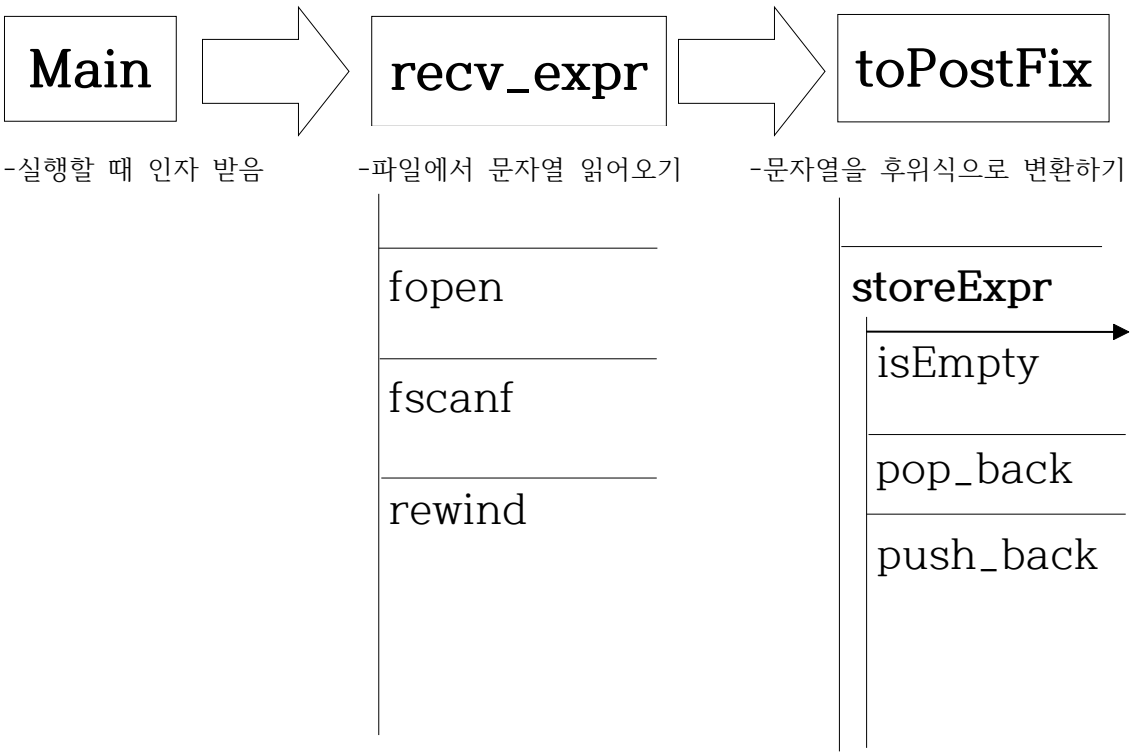
./inf_calc (파일이름)

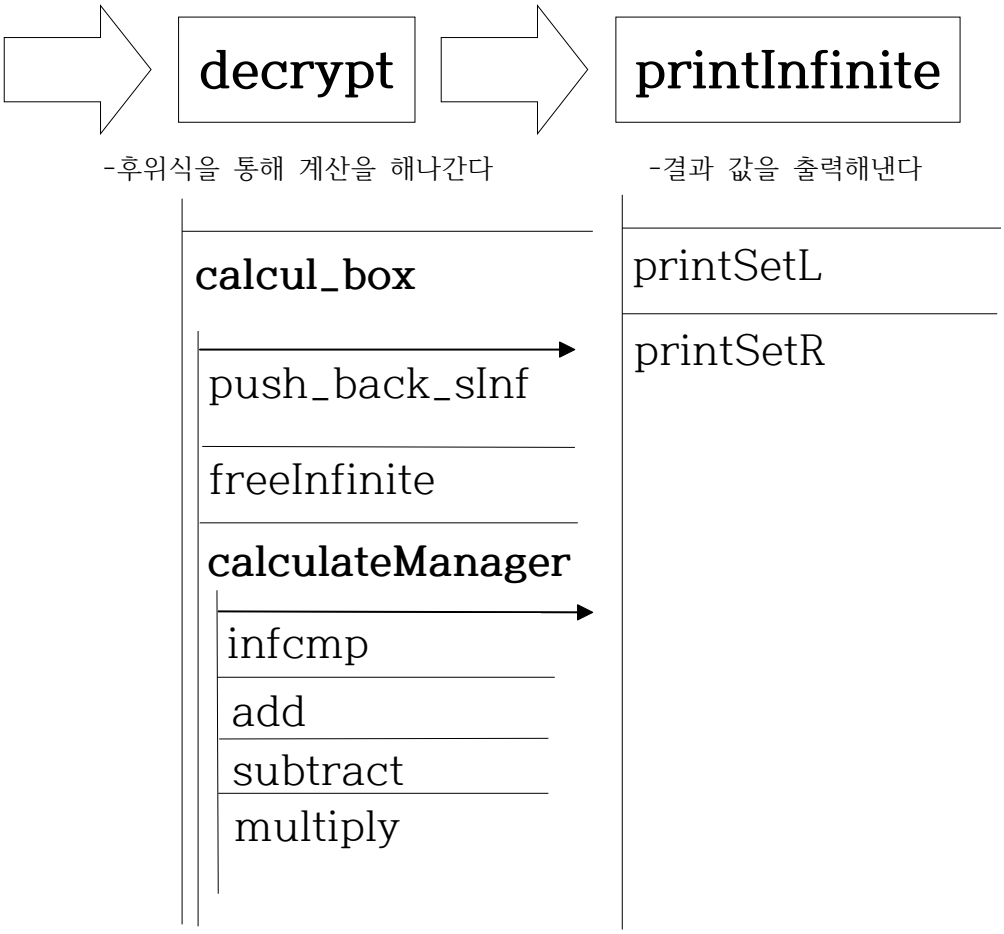
4. 테스트 방법 :

./evaluation.sh

5. 결과값 : 파일에서 계산식을 읽어온 후 계산한 값을 출력해준다.

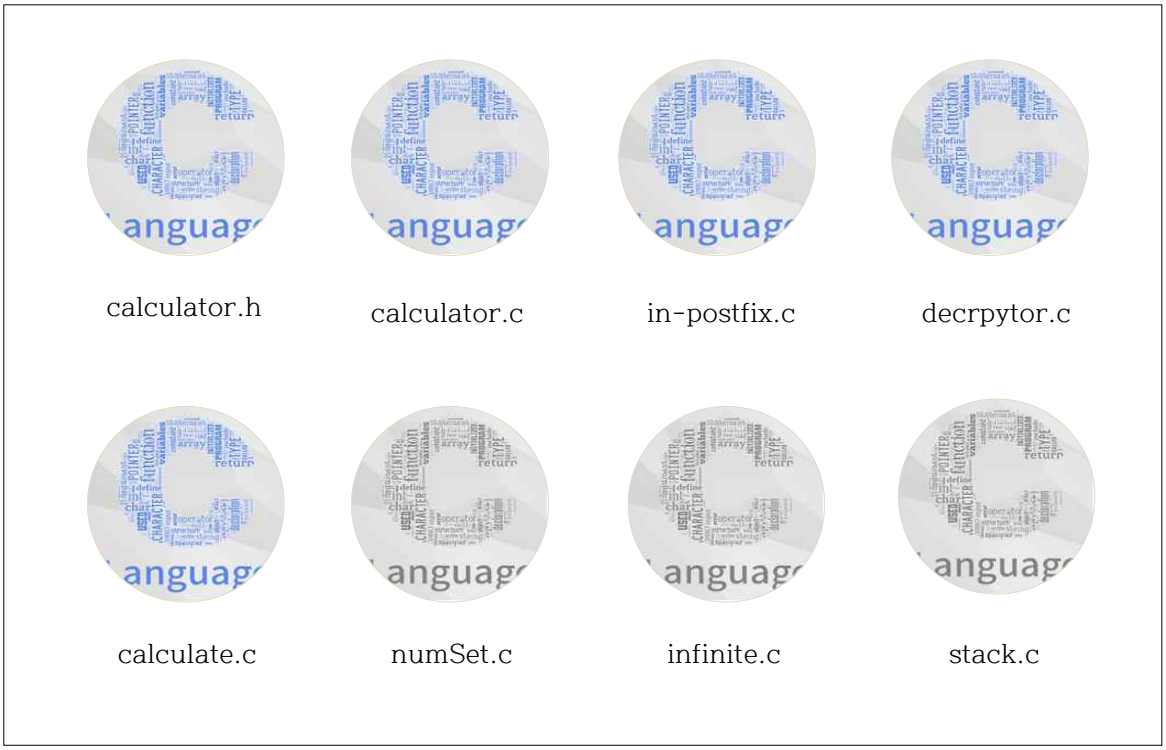
6. 전체적인 플로우차트 : **다음 페이지(Page)를 확인.**


Main
recv_expr
toPostFix
-실행할 때 인자 받음
-파일에서 문자열 읽어오기
-문자열을 후위식으로 변환하기
fopen
fscanf
rewind
storeExpr
isEmpty
pop_back
push_back


decrypt
printInfinite
-후위식을 통해 계산을 해나간다
-결과 값을 출력해낸다
calcul_box
push_back_sInf
freeInfinite
calculateManager
infcmp
add
subtract
multiply
printSetL
printSetR

0. 들어가기 전에..

파일 구성을 한번 나열해보았다.



회색 C언어 소스는 무한 수 계산에 필요한 데이터 타입을 정의하고 있다. 이러한 회색 파일들을 변수로 만들고 계산을 하며 처리를 하는 것은 **파란색** C언어 소스이다.

1. 무한 수 입력받기

1.1. char* recv_expr(char*) 함수

파일 이름(argv[1])을 recv_expr 함수에 전달해주면, 그 파일 안에 있는 내용을 char*에 정리하여 반환해준다.

1.2. 중위 -> 후위 변환 프로세스

1.2.1. char* toPostFix(char*) 함수

(1.1. char* recv_expr(char*) 함수)에서 얻은 char*를 toPostFix 에 전달한다. 그러면 연산자('+', '-', '(', ')')가 나올 때는 INSERT_OPERATOR 옵션을 전달하고 숫자가 나올 때는 INSERT_NUMBER 옵션을 전달한다. 명심해야 하는 것이 storeExpr 함수를 시작할 때, INSERT_FIRST 옵션을 전달해줘야 한다는 것이다.

1.2.2. char* storeExpr(char*) 함수

중위식을 후위식으로 바꾸는데 핵심적인 함수인데, 과정은 아래와 같다.

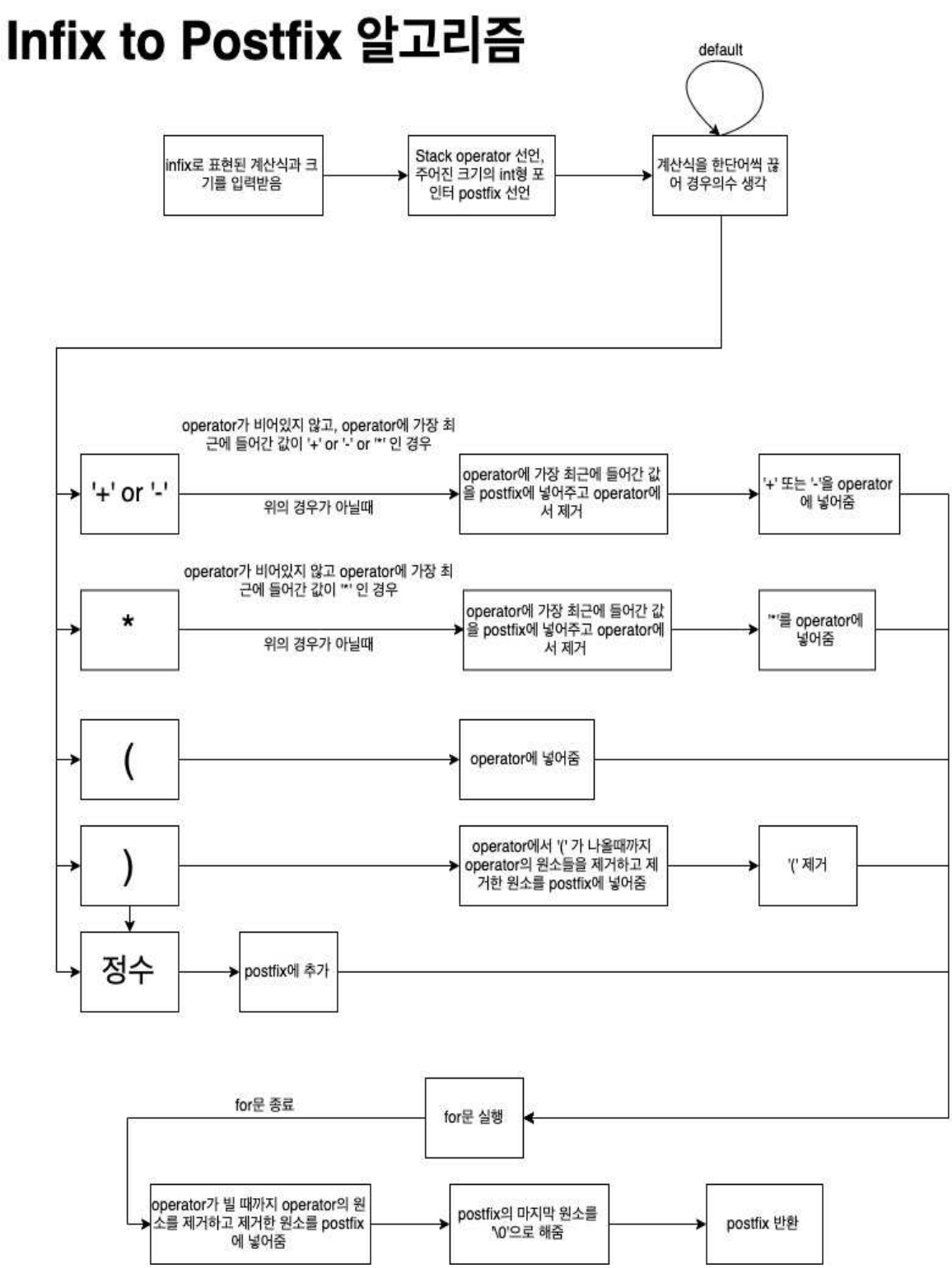


그림 1 - 중위식 -> 후위식 변환 과정 플로우차트

2. 무한 수 계산하기

2.1. 후위식 계산 프로세스

2.1.1. infinite* decrypt(char*) 함수

toPostFix 함수에서 반환된 char* 변수를 decrypt 함수에서 받는다. 이 함수에서 strtok 함수를 이용하여 숫자 또는 연산자를 받을 때마다 calcul_box에 옵션과 함께 보내준다. (decrypt 함수의 isMinus 변수를 이용해 Unary Operator를 Binary Operator로 변환한다.)

2.1.2. infinite* calcul_box(char*, int option) 함수 (storeExpr과 비슷)

calcul_box 함수는 일정 규칙으로 작동된다. 연산자가 들어오는 순간은 static stackInf* numbers 안에 있는 숫자는 항상 2개 이상이고, 바로 연산이 진행된다. decrypt 함수로부터 받은 값을 위의 내용처럼 진행하고 연산자에 맞게 calculateManager 함수를 호출한다.

2.1.2.0. infinite 클래스에 대한 설명

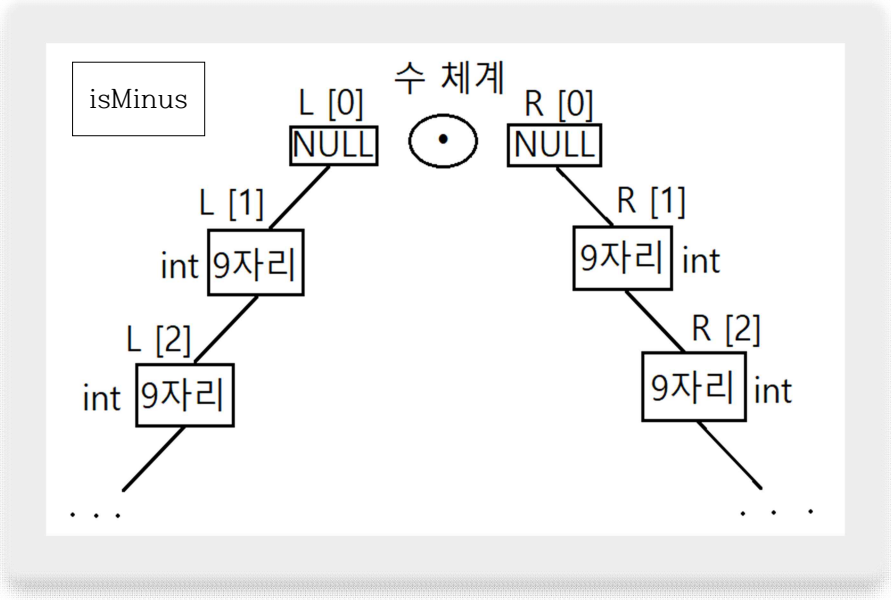


그림 2 - infinite 클래스 추상화

infinite 클래스는 점(.)을 기준으로 좌측으로 뻗어나가는 노드들과 우측으로 뻗어나가는 노드들로 구성되어 있다. 각각의 노드들은 int로 구성되어 있고, 각각 최대 9자리씩 표현할 수 있다. (왼쪽의 노드 시작점을 numSet* left라고 하고, 오른쪽의 노드 시작점을 numSet* right 라고 한다.)

isMinus 변수는 이 infinite 클래스의 값이 양수인지 음수인지 알려주는 값이다. 1이면 음수이고 0이면 양수이다.

2.1.2.1. push_back

push_back 함수는 infinite 클래스에 숫자를 삽입하는 함수이다. push_back 뒤에 R을 붙이면 right에 삽입되고, L을 붙이면 left에 삽입된다.

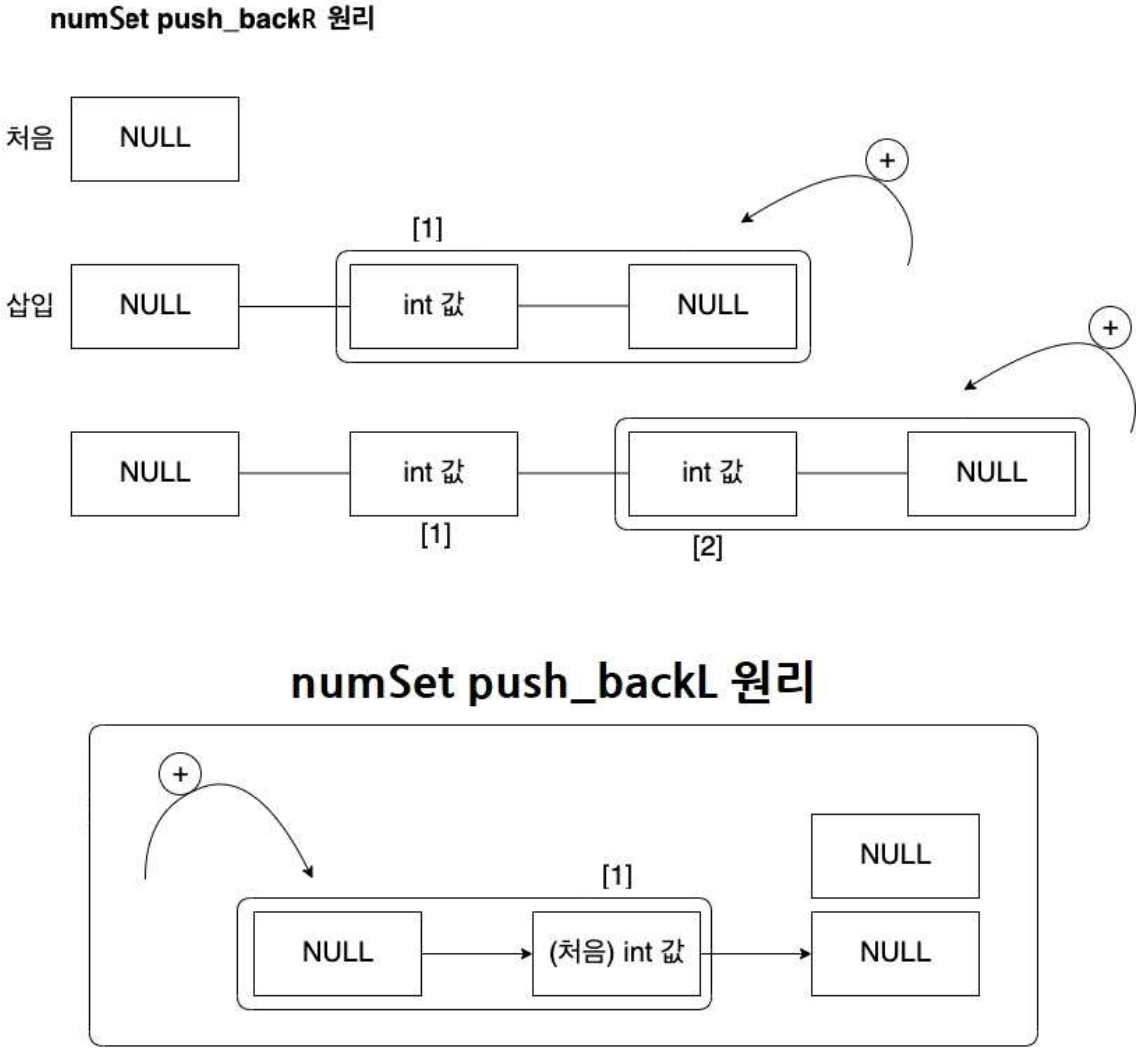


그림 3 - push_back의 원리

2.1.2.2. pop_back

pop_back 함수는 push_back의 과정을 반대로 행하는 것과 같다.

2.1.3. infinite* calculateManager(infinite*, infinite*, char) 함수

우리 조의 빼기 함수는 고질적인 문제가 있다. 바로 작은 수에서 큰 수를 못 뺀다는 것인데, 이를 해결하는 방법으로 이 함수를 만들면 좋겠다고 생각해서 구현하였다. 두 무한수의 대소비교를 하고 부호 비교를 한 뒤에 char opCode 에 맞는 계산 함수

를 호출한다. 자세한 알고리즘은 아래 코드를 보면 된다. (주석 참고)

```
61 infinite* calculateManager(infinite* first,
62         infinite* second, char opCode) {
63     infinite* newinf = NULL;
64     int fminus = first->isMinus;
65     int sminus = second->isMinus;
66     int value_of_cmp = infcmp(first, second);
67     if(opCode == '+') {
68         if(fminus == sminus) { // 부호 같다면
69             newinf = add(first, second);
70             newinf->isMinus = fminus; // fminus == sminus
71         } else if(value_of_cmp == 1) { // first > second
72             newinf = subtract(first, second);
73             if(fminus == 1) {
74                 newinf->isMinus = 1;
75             } else if(sminus == 1) {
76                 newinf->isMinus = 0;
77             }
78         } else if(value_of_cmp == -1) { // first < second
79             newinf = subtract(second, first);
80             if(sminus == 1) {
81                 newinf->isMinus = 1;
82             } else if(fminus == 1) {
83                 newinf->isMinus = 0;
84             }
85         } else {
86             newinf = initialize("0"); // 부호 다르고 값 같으면 0
87         }
88     } else if(opCode == '-') {
89         if(second->isMinus == 1)
90             second->isMinus = 0;
91         else second->isMinus = 1;
92         newinf = calculateManager(first, second, '+');
93     } else if(opCode == '*') {
94         newinf = multiply(first, second);
95         if(fminus == sminus)
96             newinf->isMinus = 0;
97         else newinf->isMinus = 1;
98     } else {
99         printf("invalid operator.\n");
00         exit(0);
01     }
```

그림 4 - calculateManager 함수

2.2.1. infinite* add(infinite*, infinite*) 함수

● 축약 패턴 정규 식

A = {0|1|2|3|4|5|6|7|8|9}$_{0+}$. {0|1|2|3|4|5|6|7|8|9}$_{0+}$

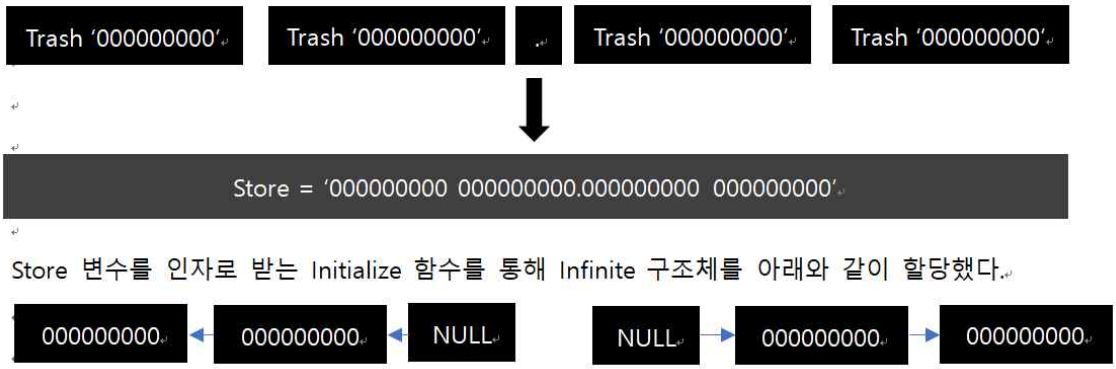
B = {0|1|2|3|4|5|6|7|8|9}$_{0+}$. {0|1|2|3|4|5|6|7|8|9}$_{0+}$

A와 B는 char* 형으로 선언, 초기화 되어 있다.

A의 패턴으로 만들어진 숫자를 점 기준으로 9자리씩 끊어서 읽는 것이 우리 조의 구조체 특징이다. 따라서, A의 점 기준 숫자의 개수가 n개 있다고 가정해보자. 그렇다면 9자리 수를 저장할 수 있는 원소를 (n % 9 == 0) ? (n / 9) : (n / 9 + 1) 개 만

큼 만들어야 한다. 그렇게 만들어진 원소 개수를 S(ize)L(eft)_A 라고 한다. 오른쪽은 S(ize)R(ight)_A 라고 한다. B도 A와 같은 방법으로 S(ize)L(eft)_B와 S(ize)R(ight)_B를 구하면 된다.

그렇다면, A와 B의 연산 결과를 저장할 새로운 Infinite 구조체를 할당해야 한다. 할당하는 방법은 결과 값의 자리 수만큼 padding값을 주어 새로운 char* 변수를 만들어 Initialize 함수를 호출한다. 가장 중요한 점은 얼마만큼 Padding값을 주어야 하냐는 것이다. 점 왼쪽으로 math.max(SL_A, SL_B) + 1 (오버플로 방지용 (+1))만큼 원소를 추가해주고, math.max(SR_A, SR_B) 만큼 오른쪽으로 원소를 추가해준다. 그렇게 만들어진 char* 변수는 store이다.



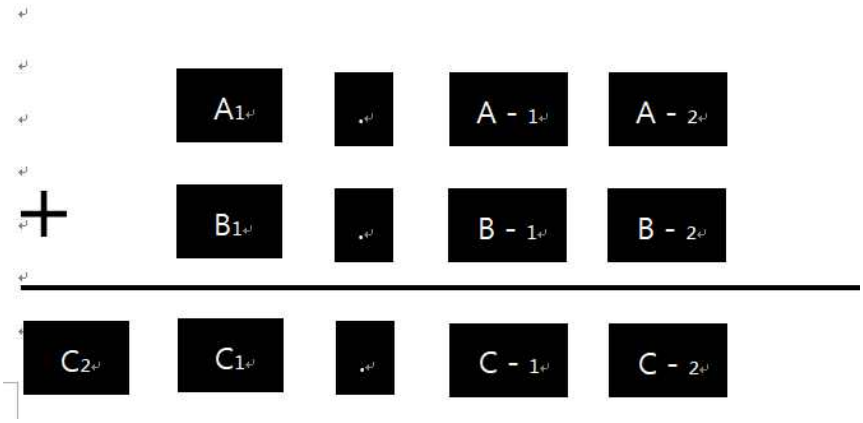
<숫자 체계 정보>

(점을 기준으로 10의 0승 자리를 1항, 10의 1승자리를 2항, … 이라고 가정하고, 10의 -1승 자리를 -1항, 10의 -2승 자리를 -2항이라고 가정한다.)

An = (n > 0 일때) 점(.) 을 기준으로 (9*(n-1) + 1)항 ~ (9*n)항 까지의 숫자를 표현한 것.

(n < 0 일때) 점(.) 을 기준으로 (9*(n+1) - 1)항 ~ (9*n)항 까지의 숫자를 표현한 것.

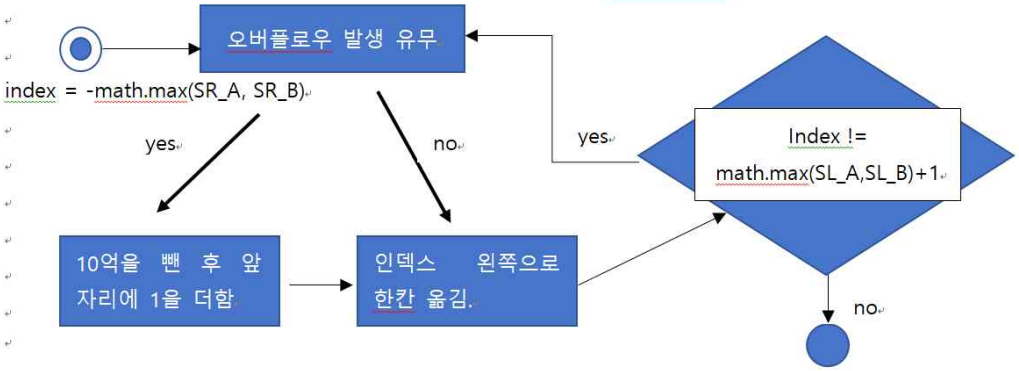
<Infinite 변수 2개를 인자로 받는 add 함수에 대한 설명>



C는 A와 B를 더한 결과값이다.

C_2는 C_1의 오버플로우를 대비하여 원소를 추가한 것이다.

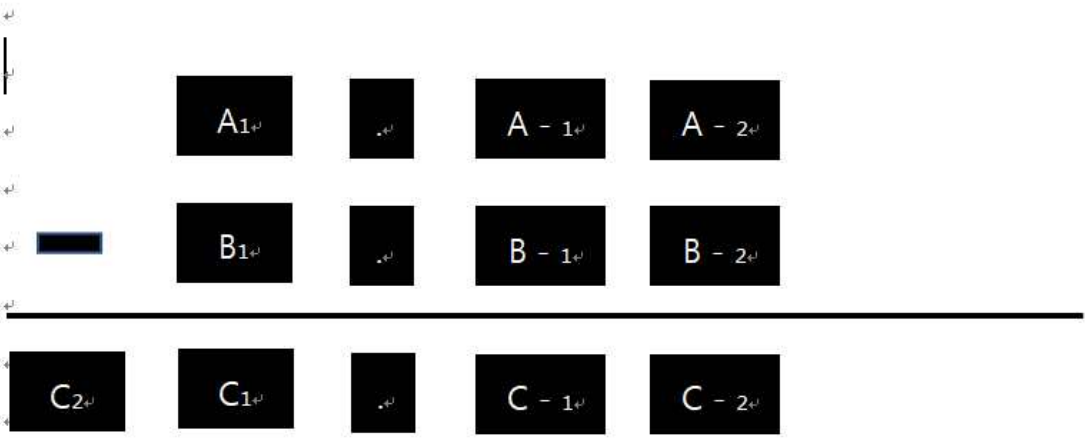
<덧셈 프로세스 플로우 차트>
Cn = An + Bn 을 BoundOutOfArray 오류에 대한 예외처리를 한 상태로 진행한다.
Cn 의 값의 결과값이 10자리가 되는지 오른쪽부터 검사한다. 플로우차트는 아래와 같다.



2.2.2. infinite* subtract(infinite*, infinite*) 함수

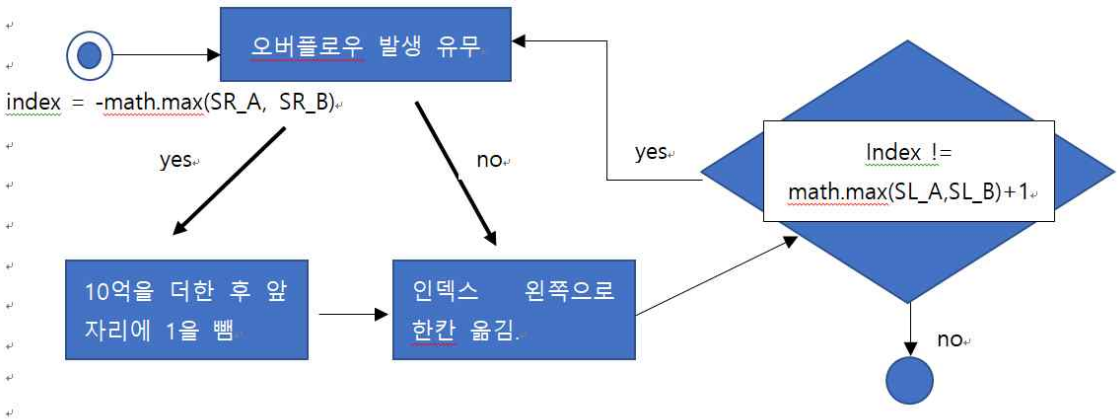
큰 수에서 작은 수 빼는 것이랑 작은 수에서 큰 수 빼는 것이 알고리즘이 달라서 무조건 큰 수에서 작은 수 빼는 것으로 조정하였다.

<Infinite 변수 2개를 인자로 받는 substract 함수에 대한 설명>



C는 A와 B를 더한 뺀값이다.

<뺄셈 프로세스 플로우 차트>
Cn = An - Bn 을 BoundOutOfArray 오류에 대한 예외처리를 한 상태로 진행한다.
Cn 값의 결과값이 음수가 되는지 오른쪽부터 검사한다. 플로우차트는 아래와 같다.



2.2.3. infinite* multiply(infinite*, infinite*) 함수

곱셈의 경우 각각의 자리수를 교차해서 곱한 다음 더하는 방식으로 연산한다. 하지만 소수점이 생기는 경우가 발생할 수 있다. 그럴 경우 소수점이 있는 수에 10*(9*n)(n은 자연수) 만큼 곱해준 후 (소수점이 있는 수)*10*(9*n) *10*(9*(-n))으로 표현한다. 이렇게 되면 N*10^(n1)과 같은 꼴을 만들 수 있고 곱해준다. 그 다음으로 자리수에 맞게 점을 찍으면 된다. 아래 표와 같은 느낌으로 2차원 배열을 통해 계산할 수 있었다.

		1	2	3
			1	2
		2	4	6
x	1	2	3	
	1	4	7	6

3. 무한 수 출력하기

3.1. void printInfinite(infinite*) 함수

왼쪽 노드들의 int 값들과 오른쪽 노드들의 int 값들을 출력해준다.

3.2. void freeInfinite(infinite**) 함수

연산이 끝난 infinite 클래스들의 메모리를 할당 해제해준다.

4. 끝

읽어주셔서 감사합니다.