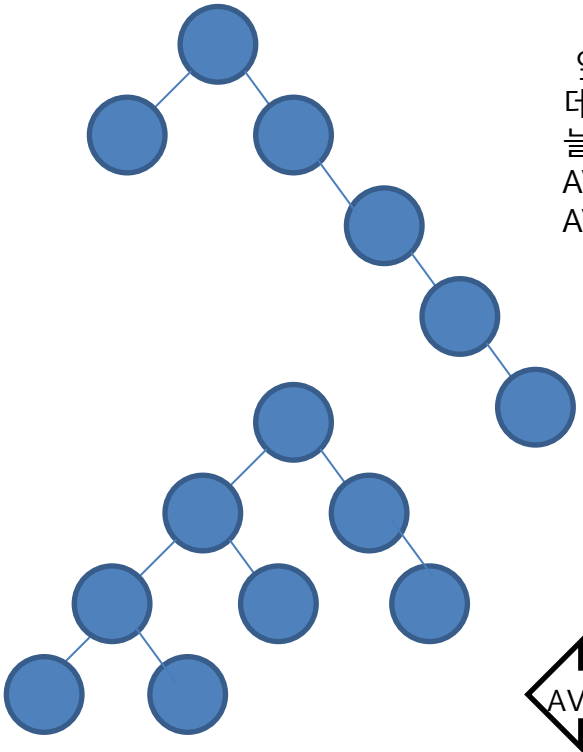
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 : Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 : 황수정
sue100012@naver.com
11일차 (2018. 03. 08)

AVL 트리(Adelson-Velskii and Landis tree)

- 한 노드를 중심으로 **좌우 종속 트리의 높이 차가 1 이하인 균형 잡힌 트리**이다.
- 이진 트리의 **삽입, 삭제를 계속할 때** 어느 한 방향으로 치우치거나, 높이 차이로 인해서 수행 시간이 증가되는 것을 막기 위해 균형을 유지하도록 한 것이다. 노드의 삽입과 삭제시에도 균형을 이룰 수 있도록 재균형이 필요하므로 삽입과 삭제가 적고, 검색이 많은 자료 구조로서 적합하다. 트리 등과 함께 균형잡힌 트리(height-balanced tree)라고도 한다.
- 이진트리가 검색을 빠르게 하기 위해서 생성된 것인데, 자료가 많아지면 그 검색의 속도 차이가 나게 된다. 이 문제점을 해결하기 위해 AVL트리가 생긴 것이다.



이진 트리 예시

AVL 트리 예시

옆의 이진 트리 예시와 같이 상대적으로 길어진 오른쪽의 데이터 검색은 왼쪽 데이터 검색 속도와 차이가 난다. 지금은 데이터가 적지만 이 것이 더 억대로 늘어난다면, 그 검색 속도의 차이는 매우 클 것이다. 그러한 문제를 해결한 것이 AVL 트리라는 것이다.

AVL 트리 예시와 같이 AVL트리는 이진 트리이면서 균형을 유지하고 있다. 균형을 유지한다는 것은 좌우 검색 높이 차이가 1이상이 없다는 것이다. 이렇게 유지 되기 때문에 검색 효율을 보장할 수 있게 된다.

그러나 이 균형이 항상 유지될 수 없다. 데이터가 쌓이다 보면 이 균형이 깨질 때가 있다. 이를 확인하기 위해 BF(Balanced Factor, 균형인자)를 사용하여 확인한다.

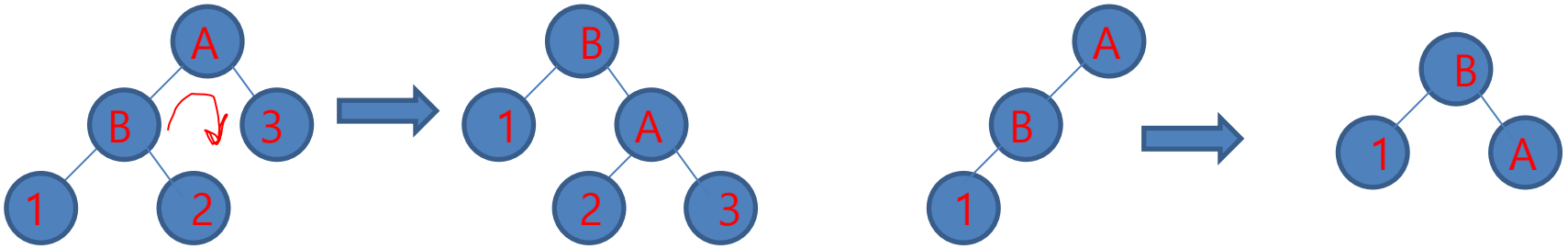
BF = 왼쪽 서브 트리의 높이 – 오른쪽 서브 트리의 높이

정상적인 BF는 -1, 0, 1을 가져야 한다. 만약 값이 2 혹은 -2 이상일 경우 트리를 변경해주어야 한다. |2|를 가졌다는 것은 한 쪽의 노드가 다른 쪽에 비해 더 많은 트리를 가졌다는 의미 이므로, 양쪽의 노드가 같은 트리값을 가져야하는 AVL 트리에 어긋난다.

보통 4가지의 경우가 있고 이를 재배치하는 과정에서 회전을 하게 된다.

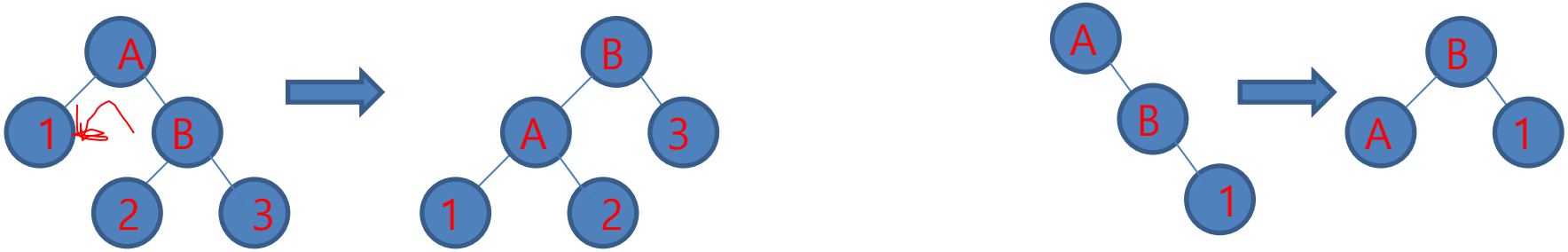
AVL 트리(Adelson-Velskii and Landis tree)

- LL
  왼쪽으로 회전시킨다고 생각하면 된다. 왼쪽 노드를 루트로 만드는 방법이다.



  A의 left자식의 left자식에 삽입되면서 문제가 발생한 경우다.

- RR
  LL의 반대로 오른쪽 노드를 루트로 만드는 방법이다. 오른쪽으로 회전시킨다고 생각하면 된다.

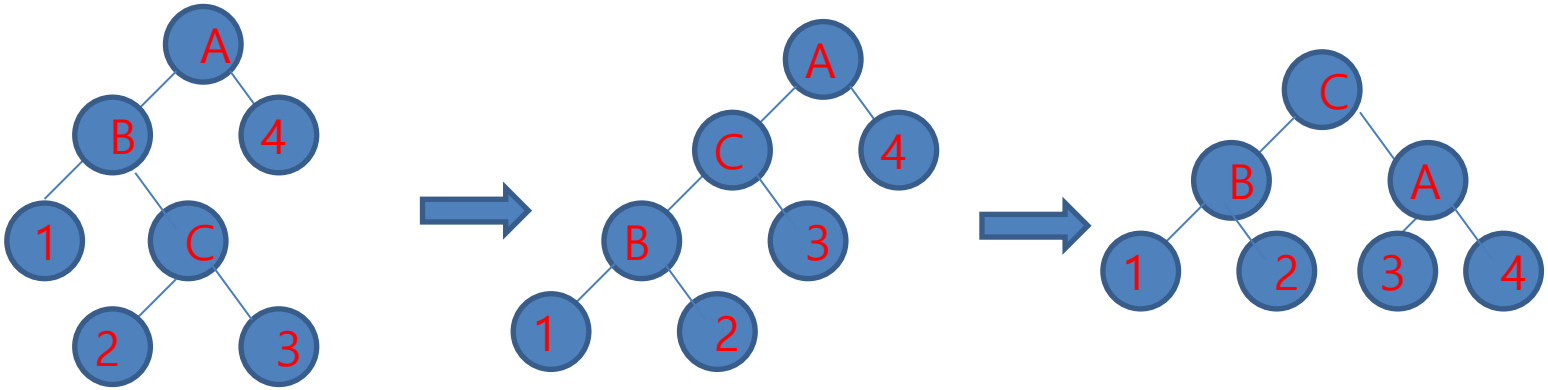


  A의 right자식의 right자식에 삽입되면서 문제가 발생한 경우다.

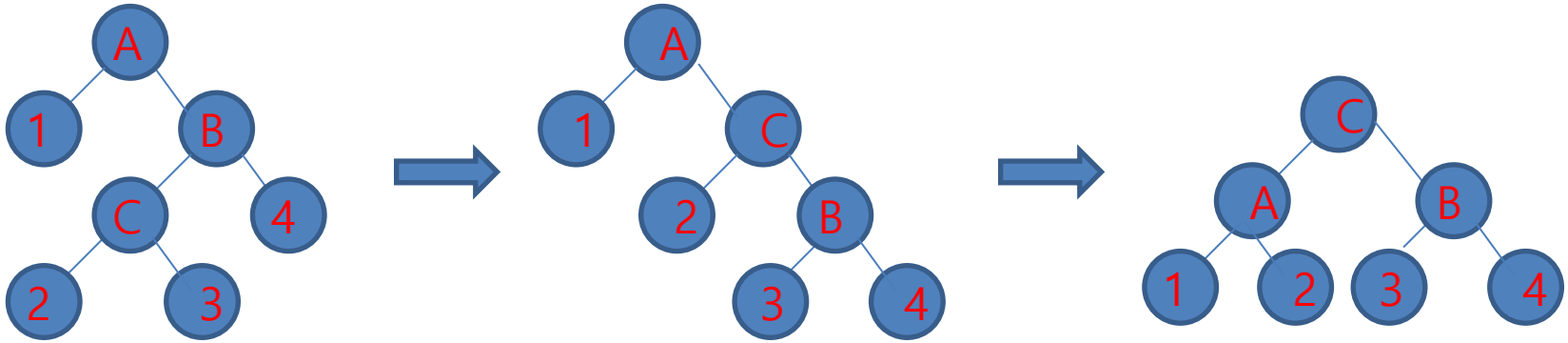
AVL 트리(Adelson-Velskii and Landis tree)

- LR

  A의 left 자식의 right자식이 삽입되면서 생긴 문제이다. 먼저 B와 C의 RR회전을 진행해 C 왼쪽 자식을 B 오른쪽 자식으로 만들고 C 왼쪽 자식에 B가 가게 한다. A, C에 대해 LL회전을 준다. C 오른쪽 자식을 A 왼쪽 자식에 두고 C 오른쪽 자식에 A가 가게 해준다.



- RL

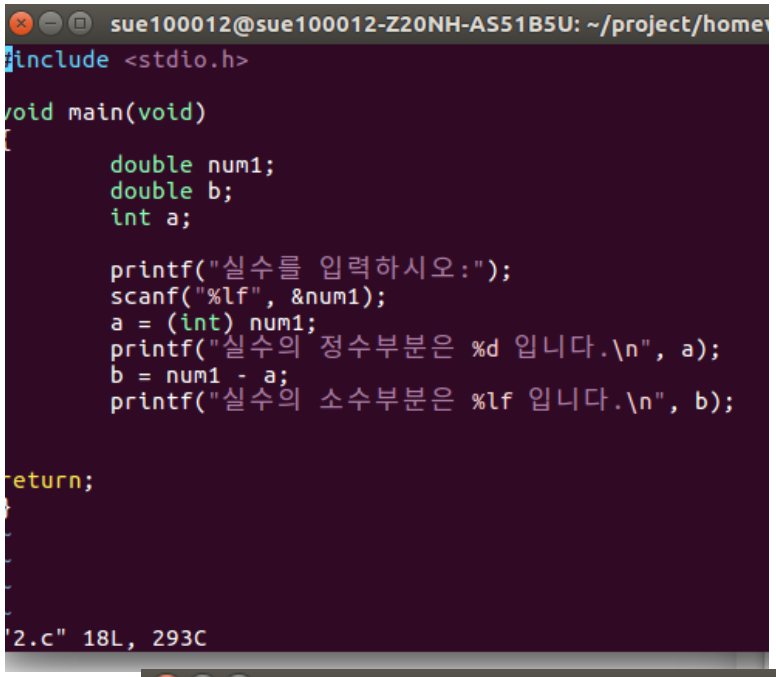


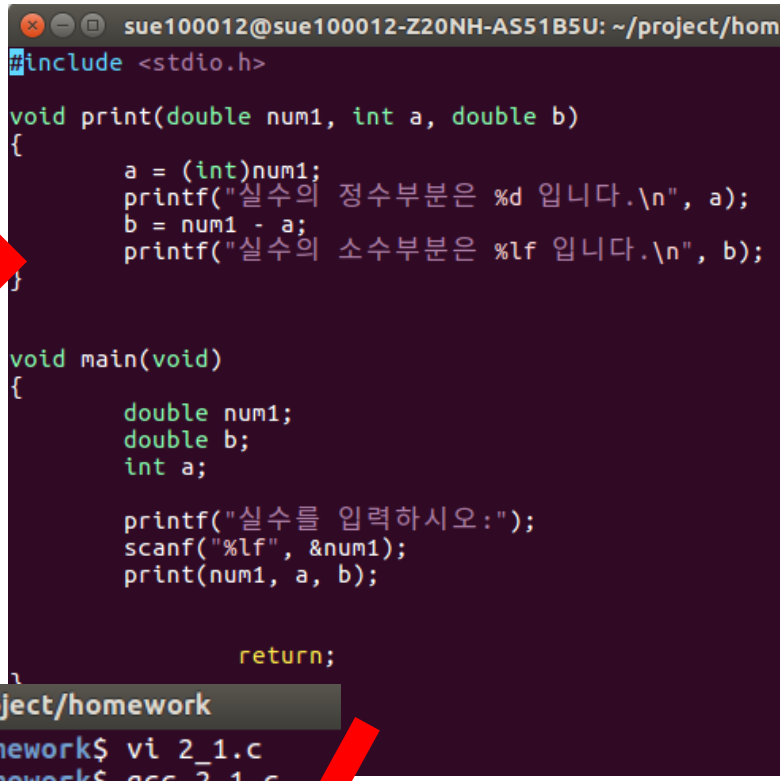
A의 right 자식의 left자식이 삽입되면서 생긴 문제이다.

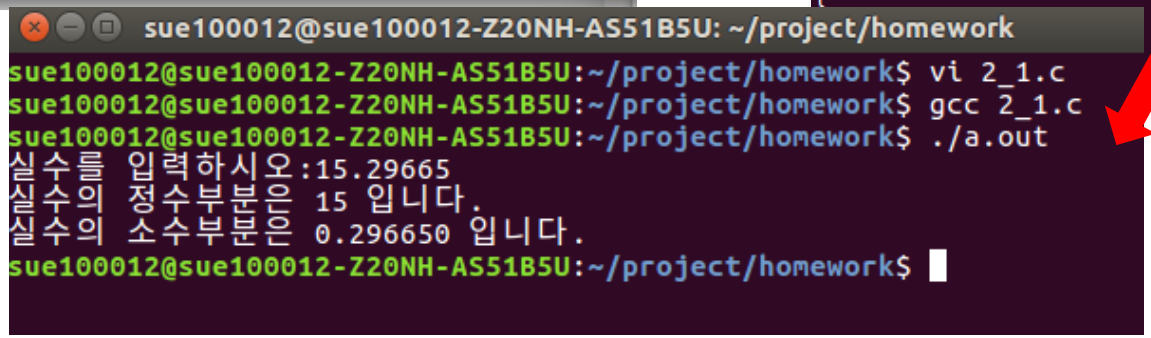
먼저 B와 C의 LL회전을 진행해서 C 오른쪽 자식을 B 왼쪽 자식으로 만들고 C의 오른쪽 자식에 B가 가게 해준다. 그리고 A, C에 대해 RR회전을 해서 C 왼쪽 자식을 A 오른쪽 자식으로 두고 C의 왼쪽 자식에 A가 가게 하는 식으로 회전을 해준다.

과제2. c언어 복습 문제

- 실수 하나를 입력받아 소수점만 분리하고 또 정수 부분만 분리하는 함수를 작성하시오. (함수 총 2개를 작성하면됨)







과제2. c언어 복습 문제

- main() 함수 안에서 인자값 2개를 가지는 doit(), letsgo() 함수를 호출하고, doit() 함수는 인자값 2개를 % 연산한 결과를 반환, letsgo() 함수는 인자값 2개를 + 연산한 결과를 반환 해서 doit()의 결과값과 letsgo()의 결과값을 곱한 결과를 출력하시오.

sue100012@sue100012-Z20NH-AS51B5U: ~/pr

```c
#include <stdio.h>

int doit(int num1, int num2)
{
int doit;
doit = num1 % num2;
return doit ;
}

int letsgo(int num1, int num2)
{
int letsgo;
letsgo = num1 + num2;
return letsgo;
}

int main (void)
{
int num1;
int num2;

num1 = 4;
num2 = 2;

printf("%d\n", doit(4, 2) * letsgo(4, 2));
return 0;
}
~
~
~
~
```

sue100012@sue100012-Z20NH-AS51B5U: ~/project/homework

```
sue100012@sue100012-Z20NH-AS51B5U:~/project/homework$ vi 4.c
sue100012@sue100012-Z20NH-AS51B5U:~/project/homework$ gcc 4.c
sue100012@sue100012-Z20NH-AS51B5U:~/project/homework$ ./a.out
0
sue100012@sue100012-Z20NH-AS51B5U:~/project/homework$
```

과제2. c언어 복습 문제

- 3과 7.7 2개의 값을 인자로 취하는 함수를 작성하고 2개를 더한 결과값을 반환할 때 정수형이 되게 하고 이를 출력해서 잘 되었는지 확인하시오.

```c
#include <stdio.h>

double sum(void)
{
int num1 = 3;
double num2 = 7.7;
double sum = 0;
sum = num1 + num2;
return sum;
}

int main(void)
{
 double sum1 = sum();
 int sum2 = (int) sum();
 printf("sum1 = %f\n", sum());
 printf("sum2 = %d\n", sum2);
 return 0;
}

~
```

```
sue100012@sue100012-Z20NH-AS51B5U:~/project/homework$ vi 3_1.c
sue100012@sue100012-Z20NH-AS51B5U:~/project/homework$ gcc 3_1.c
sue100012@sue100012-Z20NH-AS51B5U:~/project/homework$ ./a.out
sum1 = 10.700000
sum2 = 10
sue100012@sue100012-Z20NH-AS51B5U:~/project/homework$
```