

Assignment 3-3

학부: 컴퓨터정보공학부

학번: 2019202021

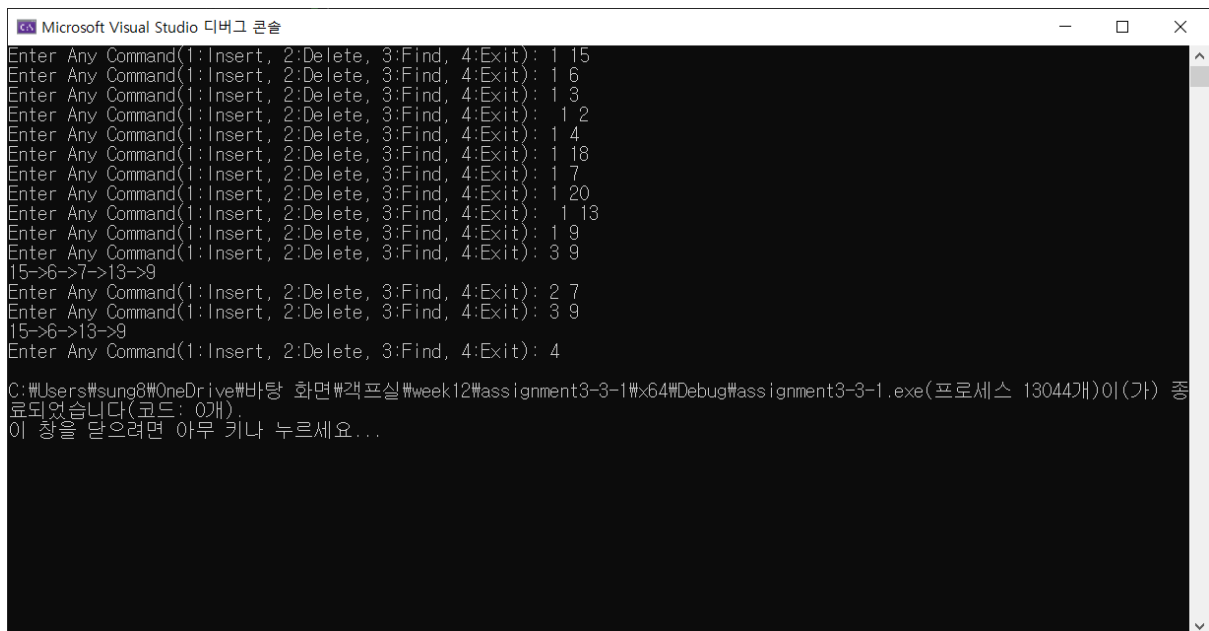
이름: 정 성 업

<1번문제>

1. 문제설명

- Binary Search Tree는 1개의 Node가 2개의 Pointer(Child)를 가져 각각의 pointer가 좌측엔 Node보다 작은 값의 Node를, 우측엔 Node보다 큰 값의 Node를 가리키는 구조로 이루어지면 최상단의 Node를 Root, Child가 없는 Node를 Leaf라고 한다. 이때 Root는 하나지만 Leaf는 여러개가 될 수 있다. Command가 1(insert)인 경우 binary search tree에 새로운 integer를 설치하고 Root부터 integer가 Node보다 작으면 좌측, 크면 우측으로 이동하며 빈 자리가 있으면 그 자리에 설치한다. Command가 2(delete)인 경우 해당 integer를 탐색하고 해당 node를 삭제한다. 이때 leaf node인 경우 leaf node만 삭제하고, child node가 1개인 경우 해당 node를 제거하고 parent node와 child node를 연결한다. 만약 child node가 2개인 경우 우측 subtree중 가장 작은 값을 해당 node값으로 바꾸고 가장 작은 값의 node를 삭제한다. Command가 3(find)인 경우 해당 node까지의 경로를 출력한다. Command가 4(Exit)인 경우 프로그램을 종료한다.

2. 결과화면



```
Microsoft Visual Studio 디버그 콘솔
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 15
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 6
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 3
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 2
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 4
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 18
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 7
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 20
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 13
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 1 9
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 3 9
15->6->7->13->9
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 2 7
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 3 9
15->6->13->9
Enter Any Command(1:Insert, 2:Delete, 3:Find, 4:Exit): 4
C:\Users\sung8\OneDrive\바탕 화면\2학기\프실\week12\assignment3-3-1\Debug\assignment3-3-1.exe(프로세스 13044개)이(가) 종
료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

3. 고찰

- Class Node에 작은 값으로 가는 Node *lower_next와 큰 값으로 가는 Node *upper_next

를 설정하고, 값을 저장할 int형 변수 num을 둔다. 노드를 생성할 경우 n값을 num값에 저장한다. Class Tree에서는 Node *root와 leaf를 두었으나 leaf가 여러 개가 될 수 있는 관계로 leaf는 명시적으로만 두었다. 이제 root가 NULL인 경우 root에 새로운 Node를 생성하고 아닌 경우 Node *currNode가 root부터 시작하여 currNode의 값보다 입력 값이 큰 경우 upper_next, 작은 경우 lower_next로 가도록 한다. 만약 currNode가 NULL이 된다면 이전 노드인 prevNode의 해당 upper_next 또는 lower_next에 새로운 노드를 생성하고 값을 저장한다.

노드를 삭제하는 경우 해당 노드를 탐색하고, 찾으면 해당 노드의 Child의 개수를 파악한다. 이 때 해당 노드가 root인 경우와 root가 아닌 경우로 나눠서 설계한다. Root가 아닌 경우에서 child가 없을 경우 이전 노드인 prevNode의 다음을 NULL로 바꾸고 currNode를 삭제한다. Child가 하나인 경우 prevNode의 다음을 currNode의 child로 연결하고 currNode를 삭제한다. 2개인 경우 currNode를 우측으로 이동하고 좌측 child 노드가 있을 때까지 이동한다. 없을 경우 해당 currNode가 가장 작은 값이고 바꿀 노드와 값을 바꾸고 currNode의 child가 없을 경우 prevNode의 다음을 NULL로, 있을 경우 prevNode의 다음을 child로 바꾼다. 이때 삭제할 currNode의 lower_next가 없으므로 2가지만 고려하는 것이다. 그 이후 currNode를 삭제한다.

노드를 찾는 경우 해당 노드 값과 현재 노드 값을 비교해 가며 출력하며 이동한다. 값을 찾은 경우에는 " -> "를 출력하지 않는다.

마지막으로 프로그램을 종료하는 경우 new Node로 만들어낸 노드를 삭제해야하므로 root부터 시작하여 현재 노드가 nullptr이 아닌 경우 우측 child와 좌측 child를 삭제하는 함수를 부르는 재귀함수를 통해 삭제한다.

<2번문제>

1. 문제설명

- 지하철의 최단거리를 계산하는 프로그램을 구현하는 것으로, 프로그램은 Assignment3-3-2-1.txt에서 1호선 데이터를 Assignment3-3-2-2.txt에서 2호선 데이터를 읽어온다. 1호선 노드와 2호선 노드는 Doubly Linked List로 연결되어 있고, 추가적으로 2호선은 Head와 Tail이 연결된 Doubly Circular Linked List이다.

프로그램은 출발역과 도착역을 입력받고 가장 적은 역을 거쳐 도착할 수 있는 경로를 출력하도록 하고 환승은 시청역에서만 가능하다. 1호선 데이터와 2호선 데이터는 과제에 주어진 텍스트파일을 사용한다.

2. 결과화면

```
Microsoft Visual Studio 디버그 콘솔
출발역 : 노천
도착역 : 이대
노천->월계->광운대->석계->신이문->외대앞->회기->청량리->제기동->신설동->동묘앞->동대문->종로5가->종로3가
->종각->시청->충청로->아현->이대
C:\Users\sung8\OneDrive\바탕 화면\책\프실\week12\assignment3-3-2\64\Debug\assignment3-3-2.exe(프로세스 1
1084개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

3. 고찰

- 이전까지의 assignment에서 사용한 텍스트파일을 읽어오는 방법을 그대로 사용한다. Class Node는 양방향이므로 Node*next와 Node*prev를 선언하고 역이름을 저장할 변수 char name[30]을 선언한다. 노드를 생성할 경우 저장할 이름값, 다음 노드, 이전 노드 모두 가져와 저장하도록 한다.

1호선은 이전에 해본 양방향 노드 생성을 사용하였으나 2호선은 양방향 노드 생성과 함께 head와 tail을 연결해야 했다. 그래서 2호선 노드를 생성할 때마다 마지막 노드는 무조건 다음을 NULL이 아닌 head를 가리키도록 했고 이전 노드는 마지막 노드를 가리키도록 했다. 그리고 head의 이전은 마지막 노드를 가리키도록 하여 양방향 연결과 원형 Linked List를 완성했다.

경로를 찾을 경우 출발과 도착이 1호선만 있는 경우, 2호선만 있는 경우, 1호선 출발하고 2호선 도착인 경우, 2호선 출발하고 1호선 도착인 경우 4가지로 보았고 환승은 어차피 '시청'에서 하므로 호선이 바뀐 경우 해당 호선 출발부터 '시청'까지, 다른 호선 '시청'부터 도착역까지 경로를 출력하되 다른 호선에서 '시청'은 출력하지 않도록 한다. 최단 경로 출력은 1호선은 출발역의 num1이 도착역의 num2보다 큰 경우 prev로 이동하고, 작은 경우 next로 이동하며 출력한다. 2호선은 원형이므로 출발역의 num1과 num2를 크기를 비교하되 2호선 전체 역의 개수에서 num1을 빼고 num2를 더한 값과 num1에서 num2를 뺀 값을 비교하거나, 전체 역 개수에서 num2를 빼고 num1을 더한 값을 num2에서 num1을 뺀 값을 비교하며 최단 거리를 찾았다.

<3번문제>

1. 문제설명

- 정해진 주문을 요리사에게 할당해 요리하는 시뮬레이터 프로그램을 구현하시오. 프로그램은 시작 시 제공받은 Assignment3-3-3-1.txt를 읽어 3명의 요리사(Jack,Bob,John)가 각 요리에 걸리는 시간정보를 저장하고 Assignment3-3-3-2.txt로부터 주문을 읽어 주문마다 어떤 요리사가 요리할지 사용자로부터 입력받아 각 요리사의 Queue에 저장한다.

이 때 입력 받은 요리사가 할 수 없는 요리면 재 입력을 요청하고 3명의 요리사 모두가 할 줄 모르는 요리는 주문이 오지 않는다고 가정한다.

입력이 끝나면 각 요리사는 프로그램 예시와 같이 Queue에 저장된 주문을 요리하며 매 시간마다 어떤 요리를 하고 있는지 현황을 출력한다. 모든 요리사가 요리를 끝내면 얼마나 걸렸는지 출력하고 프로그램은 종료된다.

2. 결과화면

```
Microsoft Visual Studio 디버그 콘솔
New order : pizza
Pick Cooker : Jack
New order : rawfish
Pick Cooker : John
New order : frenchfries
Pick Cooker : Bob
New order : pasta
Pick Cooker : Bob
New order : takoyaki
Pick Cooker : John
New order : coke
Pick Cooker : Jack
New order : steak
Pick Cooker : John
New order : frenchfries
Pick Cooker : Bob

Cooking(pizza1/8)      John      Cooking(rawfish2/3)      Bob      Done!(frenchfries)
Cooking(pizza6/8)      Cooking(rawfish1/3)      Cooking(pasta3/4)
Cooking(pizza5/8)      Done!(rawfish)      Cooking(pasta2/4)
Cooking(pizza4/8)      Cooking(takoyaki4/5)      Cooking(pasta1/4)
Cooking(pizza3/8)      Cooking(takoyaki3/5)      Done!(pasta)
Cooking(pizza2/8)      Cooking(takoyaki2/5)      Done!(coke)
Cooking(pizza1/8)      Cooking(takoyaki1/5)      Done!(frenchfries)
Done!(pizza)      Done!(takoyaki)
Cooking(coke2/3)      Cooking(steak4/5)
Cooking(coke1/3)      Cooking(steak3/5)
Done!(coke)      Cooking(steak2/5)
Cooking(takoyaki3/4)      Cooking(steak1/5)
Cooking(takoyaki2/4)      Done!(steak)
Cooking(takoyaki1/4)
Done!(takoyaki)

Total Time is :15
C:\Users\chung\OneDrive\바탕 화면\프실\week12\Assignment3-3-3\Debug\Assignment3-3-3.exe (프로세스 9072개)이 (가) 종료되었습니다(코드: 0x0).
이 창을 닫으려면 아무 키나 누르세요 ...
```

3. 고찰

- 텍스트파일을 읽어오는 것은 2번 문제와 같이 하면 되며, 문자열을 끊어서 읽어오는 기준은 공백 또는 개행이다. 다만 이번 문제에서 고생했던 점은 다운받은 Assignment3-3-3-1.txt파일의 파일명이 Assignemnt3-3-3-1.txt로 되어있어서 코딩 오류로 읽어오지 못하는 것이라 생각하여 여러 번 다시 코드를 확인했으나 결국 문제는 파일명이 잘 못된 것이었다.

클래스는 4가지로 잡아 1개는 노드를 나타내며 음식정보와 시간을 기록한다. 다른 3개는 각각 Jack, Bob, John의 queue를 관리하는 클래스이고 주문된 음식 정보 입력과 현재 주문의 진행도 출력, 그리고 마침의 확인하는 함수 3개로 이루어져 있다. 이때 주문한 음식의 진행도 출력하는 함수는 함수에서 출력하는 것이 아닌 return을 해당 문자열로 반환한다. 이를 위해 반환될 문자열을 static char 배열로 선언하여 준다. 추가적으로 이번 클

래스는 소멸자에서 노드를 삭제하는 것이 없는데 이는 출력하면서 완료된 노드는 바로 삭제를 진행하였기 때문이다.

메인에서는 각 요리사와 가능 요리, 시간을 텍스트파일로부터 입력 받고 각 배열에 저장한다. 그리고 후에 주문이 들어왔을 때 해당 요리사가 가능여부를 bool available_요리사 3개로 각각 관리하여 못하는 요리일 경우 다시 물어보도록 했다. 마지막으로 출력에서 모든 출력이 마무리되었을 때 반복문을 탈출하고 가운데 정렬을 위해 출력할 문장의 길이를 가져와 전체 길이에서 빼고 이를 2로 나누어 먼저 공백을 해당 길이만큼 출력하고 나머지 길이를 왼쪽 정렬하여 출력했다. 또한 반복문이 진행될 때마다 카운트하여 후에 총 시간을 계산하였다.

<4번문제>

1. 문제설명

- 학생정보를 관리하는 프로그램을 구현하는 것으로 전공, 학번으로 이루어진 Linked List 들로 구현되고 학번으로 이루어진 Node마다 이름을 저장하는 Binary Search Tree를 가진 구조로 학생정보를 저장한다. 모든 구조는 숫자는 오름차순, 문자는 사전식 정렬을 따른다.

프로그램 시작 시 Assignment3-3-4.txt로부터 학생정보를 읽어와 자료구조를 구축하고 Command에 따라서 동작한다. Command가 1(insert)인 경우 삽입 경로를 출력하고 이를 저장한다. 이때 정보가 모두 같은 학생은 없다고 가정한다. 2(delete)인 경우 해당 정보와 일치하는 학생정보를 제거하고 제거 후 학번에 해당하는 학생이 없는 경우 학번 Node를 제거하고 전공 내 학번이 없을 경우 전공 Node를 제거한다. 3(Print_all)인 경우 모든 학생 정보를 순서와 무관하게 출력하도록 하고, 4(Print_major)인 경우 입력 받은 해당 전공의 모든 학생 정보를 출력한다. 이때도 순서는 무관하며, 5(Print_year)인 경우도 입력 받은 학번의 모든 학생 정보를 순서 무관하게 출력한다. Command가 6(Print_Name)인 경우 이름과 같은 학생을 찾아 탐색 경로와 함께 출력을 하며, 이름이 같은 학생을 발견할 때마다 학생을 찾은 경로를 출력한다. 마지막으로 Command가 7(Exit)인 경우 프로그램을 종료한다.

2. 결과화면

```
C:\Users\sung8\OneDrive\바탕 화면\프젝트실\week12\assignment3-3-4\64\Debug\assignment3-3-4.exe
Enter Any Command(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_id, 6:Print_Name, 7:Exit): 3
Chemistry 2018 Sophia
Chemistry 2018 Isabella
Chemistry 2018 Emma
Chemistry 2018 Olivia
Chemistry 2018 Mia
Chemistry 2018 Ava
Chemistry 2019 Emily
Chemistry 2019 Noah
Chemistry 2019 Liam
Chemistry 2019 Mason
Chemistry 2019 Jacob
Chemistry 2020 William
Chemistry 2020 Ethan
Chemistry 2020 Michael
Chemistry 2021 Alexander
Chemistry 2021 Jayden
Chemistry 2021 Madison
Chemistry 2021 Daniel
Chemistry 2021 Elizabeth
Chemistry 2021 Charlotte
Chemistry 2021 Aislin
Computer_Information_Engineering 2021 Jackson
Computer_Software 2016 Avery
Computer_Software 2016 Sofia
Computer_Software 2016 Chloe
Computer_Software 2018 Ella
Computer_Software 2018 Harper
Computer_Software 2018 Anella
Computer_Software 2018 Aubrey
Computer_Software 2018 Elijah
Computer_Software 2020 Aiden
Computer_Software 2020 James
Computer_Software 2020 Matthew
Computer_Software 2020 Benjamin
Electrical_Engineering 2015 Zoe
Electrical_Engineering 2018 Audrey
Electrical_Engineering 2018 Leah
Electrical_Engineering 2018 Allison
Electrical_Engineering 2021 Anna
Electrical_Engineering 2021 Logan
Electrical_Engineering 2021 David
Electrical_Engineering 2021 Anthony
Electrical_Engineering 2021 Aaliyah
Physics 2021 Samuel
Physics 2021 Christopher
Physics 2021 John
Physics 2022 Dylan
Physics 2022 Isaac
```

```
Microsoft Visual Studio 디버그 콘솔
Enter Any Command(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_id, 6:Print_Name, 7:Exit): 4 Computer_Information_
Engineering
Computer_Information_Engineering 2021 Jackson
Enter Any Command(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_id, 6:Print_Name, 7:Exit): 1 Computer_Information_
Engineering 2022 Tim
Chemistry -> Computer_Information_Engineering -> 2021 -> 2022 -> Tim
Enter Any Command(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_id, 6:Print_Name, 7:Exit): 2 Computer_Information_
Engineering 2022 Tim
Delete_Name : Tim
Delete_Name : 2022
Enter Any Command(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_id, 6:Print_Name, 7:Exit): 2 Computer_Information_
Engineering 2021 Jackson
Delete_Name : Jackson
Delete_Name : 2021
Delete_Major : Computer_Information_Engineering
Enter Any Command(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_id, 6:Print_Name, 7:Exit): 6 Benjamin
Chemistry -> Computer_Software -> 2016 -> 2018 -> 2020 -> Aiden -> James -> Benjamin
Enter Any Command(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_id, 6:Print_Name, 7:Exit): 5 2020
Chemistry 2020 William
Chemistry 2020 Ethan
Chemistry 2020 Michael
Computer_Software 2020 Aiden
Computer_Software 2020 James
Computer_Software 2020 Matthew
Computer_Software 2020 Benjamin
Enter Any Command(1:Insert, 2:Delete, 3:Print_all, 4:Print_major, 5:Print_id, 6:Print_Name, 7:Exit): 7
C:\Users\sung8\OneDrive\바탕 화면\프젝트실\week12\assignment3-3-4\64\Debug\assignment3-3-4.exe (프로세스 8672개)이 (가) 종료되
었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

3. 고찰

- 이번 assignment3-3의 제일 어려운 문제는 4번문제로 입력과 찾기 등은 이전 과제 그
리고 이번 1번 문제를 참고하며 비교적 쉽게 구현하였다. 다만 한번에 전공, 학번, 이름을
Node로 설정하는 경우 한 함수의 코드가 길어질 뿐더러 오류가 날 경우 바로 어디 함수
의 문제인지 파악하기 어려웠기에 각 노드를 생성하는 함수를 따로 만들어 노드를 만들

도록 했다. 또한 파일의 입력인지 또는 메인에서의 입력인지를 구분하여 파일로부터의 입력일 경우 입력 경로를 출력하지 않도록 했다. 다만 처음에 오류가 났는데 이는 생각보다 긴 전공의 이름을 생각 못하고 배열의 길이를 짧게 잡았기 때문이다.

해당 노드 삭제는 제일 오래 걸렸다. 1번 문제 같은 경우 숫자를 바로바로 비교할 수 있었으나 4번 문제 같은 경우 문자열의 문자 하나하나 크기를 비교해 가며 이동하고, 또한 삭제된 노드와 이전 노드의 관계를 가져와 NULL로 연결하거나 다음 노드로 연결하여 겹치는 노드가 없도록 하여 Binary Search Tree 연결이 제대로 작동하도록 했다. 그리고 이름 같은 경우 첫 노드가 삭제 되는 경우 yearNode의 다음에 새롭게 연결해야 하므로 이를 고려하여 첫 노드인 경우 그리고 아닌 경우로 두었다. 해당 노드 탐색 및 삭제를 완료한 후에는 삭제한 이름을 출력하도록 했고 해당 학번의 name_next가 없는 경우 yearNode를 삭제하고 이전 노드를 NULL 또는 다음 yearNode와 연결했다. 만약 전공의 yearNode가 없는 경우 majorNode를 삭제하고 이전 노드를 NULL 또는 다음 majorNode와 연결했다.

전공, 또는 학번, 또는 이름을 찾을 경우 currNode를 이동하여 해당 정보를 찾을 때까지 이동하고 찾은 경우 출력하도록 했다. 전공을 출력하는 경우는 1번 문제 소멸자처럼 재귀함수를 통해 모든 Binary Search Tree를 출력하도록 했다. 이름을 찾을 경우 해당 전공, 학번, 이름 노드를 모두 print_path_info에 넘겨 새로운 currNode가 이를 찾을 때까지 경로를 출력하도록 한다.

이미 정상 작동하기에 더 만지진 않았지만 노드를 삭제할 때 계속 이전 노드와 비교할 것이 아닌 이전 노드와 현재 노드의 관계를 어느 변수에 저장해두었으면 더 간단하지 했을 것 같더라는 생각을 했다.