<자료구조 1차 프로그래밍 과제 보고서>

소프트웨어학과 201720736 이균

1. 문제 이해

고객의 이름, 고객의 유형, 고객의 핸드폰 번호, 구매/판매 금액이 입력으로 들어오는 것으로 보아 구조체를 통한 자료형의 효율적인 표현을 해야 한다.

또한 고객의 유형이 판매자, 구매자 두 가지 유형이 있으므로 두 가지 유형을 구분해야 각각의 구매/판매액 중에서 최대, 최솟값이 무엇인지, 또한 평균은 무엇인지 계산할 수 있다.

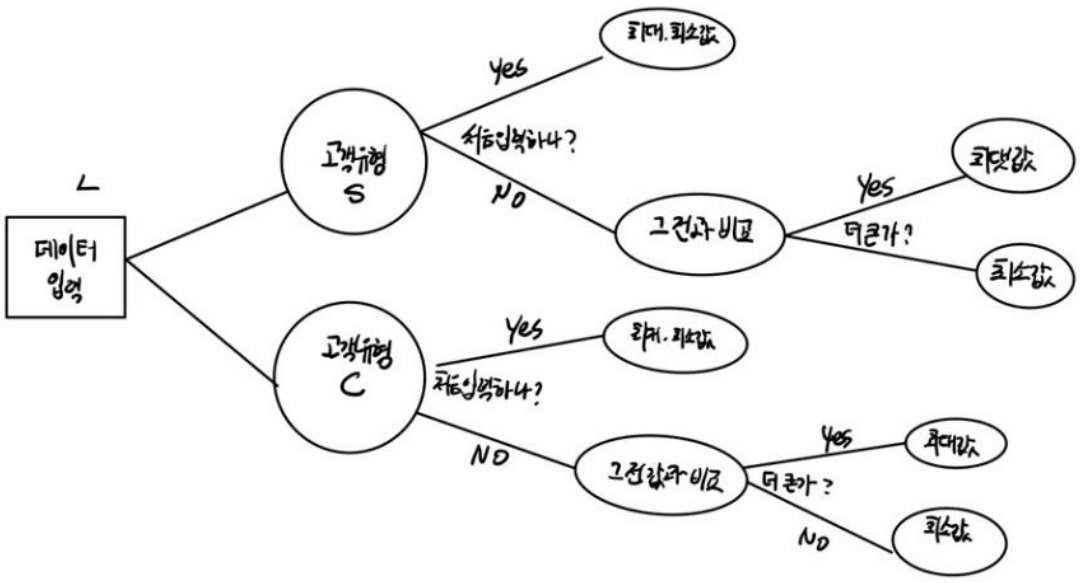
또한 입력이 100명 이하로 이루어지기 때문에 구조체 배열을 사용해야 할 것이다.

2. 고객을 위한 데이터

```
typedef struct user
{
    char name[20];
    enum tagfield {S, C} utype;
    char phoneNum[20];
    union {
        int smoney;
        int cmoney;
    }money;
}User;
```

- User에는 고객의 이름을 담을 수 있는 배열이 있다.
- enum을 활용하여 utype변수를 선언하여 고객의 유형 즉 S(seller), C(customer)을 구분하였다.
- 핸드폰 번호를 배열에 저장하였다.
- 고객의 유형에 따른 금액을 union을 이용하여 표현하였다.
- main함수에서는 구조체 배열을 통해 100명 이하의 고객에 대한 정보를 담을 수 있도록 하였으며 입력 받은 데이터는 구조체 배열의 각 구조체 변수와 Union변수에 담기게 된다.

3. Flowchart



프로그램에서 입력 받은 데이터의 고객유형이 무엇인지에 따라 그 전에 입력 받은 값과 현재의 값을 계속 비교해 나가며 최대 최소값을 계산한다. 만약 입력이 처음이라면 그 값은 최솟값 또는 최댓값일 것이다.

4. 알고리즘

최저/최대 구매자/판매자를 계산하기 위해서 다음과 같은 알고리즘을 사용하였다.

- 최대 구매자/판매자 계산을 위한 코드 구성

  구매자냐 판매자냐만 다를 뿐 코드의 구성이나 알고리즘은 동일하기 때문에 고객 유형 S인 경우를 설명하겠다.

```
if (arr[i].utype == S) // 판매자의 최댓값을 구하기 위한 조건문
{
    if (i == 0)//맨처음입력
    {
        MaxSmoney = arr[i].money.smoney;
        MaxSindex = 0;
    }
    else
    {
        if (MaxSmoney <= arr[i].money.smoney) // 최댓값이 같을 경우도 인덱스를 최신화 해서 가장 최근 고객을 출력
        {
            MaxSmoney = arr[i].money.smoney;
            MaxSindex = i;
        }
    }
}
```

우선 고객 형식을 S또는 C로 비교한 후 그에 부합할 경우 최대 구매자/판매자를 계산하기 위해서 처음 입력으로 들어온 금액의 경우에는 무조건 최대값이 되기 때문에 그것을 위한 조건문과, 처음 입력 이후에 더 큰 금액이 들어올 경우 최댓값을 바꾸는 형식으로 이루어져 있다.

또한 만약 같은 금액이 들어온다면 인덱스를 최신화 하여 추후 출력에서 최신의 인덱스를 출력하면 가장 최근 입력된 사람이 나오게 된다.

- 최저 구매자/판매자 계산을 위한 코드

```
if (arr[i].utype == S) // 판매자의 최솟값을 구하기 위한 조건문
{
    if (i == 0)
    {
        MinSmoney = arr[i].money.smoney;
        MinSindex = 0;
    }
    else
    {
        if (MinSmoney != 0 && MinSmoney >= arr[i].money.smoney)  // 처음 입력이 들어와서 MinSmoney가 바뀌었거나 또는 다음 입력에서 더 작은
        {
            MinSmoney = arr[i].money.smoney;
            MinSindex = i;
        }
        else if (MinSmoney == 0) // 처음 입력이 S가 아닌 C일 경우 MinSmoney는 여전히 0이므로 최신화해야 함
        {
            MinSmoney = arr[i].money.smoney;
            MinSindex = i;
        }
    }
}
```

앞서 최댓값을 구할 때와 마찬가지로 처음 입력이 들어올 경우 최솟값이 되고, 그 이후 입력이 들어올 경우에는 처음 입력한 값과 비교해서 최솟값을 결정하면 되는데 이때 특이점은 처음 예를 들어 처음 입력이 C이고 S의 입력이 없었을 때, 그 다음(두번째) 입력으로 S가 들어올 경우, MinSmoney는 여전히 0이지만 이는 다음에 들어온 값보다 무조건 작기 때문에 오류가 발생한다. 따라서 i가 0이 아닐 경우에서 케이스를 MinSmoney가 0과 0이 아닐 때로 구분한다. 이러한 과정을 거쳐서 입력이 없다가 나중에 입력이 들어온 경우 최솟값을 정상적으로 계산할 수 있게 되고, 그 이후 또 데이터가 추가된다면 이전 금액과 비교해서 더 작거나 같은 값을 최솟값으로 한다. (최신의 입력을 최솟값으로 출력해야 하므로 등호 포함)

- 평균 계산을 위한 코드

평균 계산은 데이터가 들어올 때 마다 cnt를 증가시키고 고객 유형에 따른 금액의 총 합을 cnt로 나누면 된다.

- 출력

출력에서 데이터의 입력이 하나도 없다면 (none)으로 출력하고 아니면 정상적으로 최고/최저 금액과 평균액을 둘째자리에서 반올림하여 출력한다.

5. Union의 의미

과제코드에서 union은 사용자유형에 따른 금액을 나타내기 위해 사용하였는데, 만약 단순히

구조체 안에 int money만 선언했다면 이후 코드에서 money가 Seller의 money인지, Customer의 money인지 코드의 가독성이 매우 떨어질 것이다. 즉 최댓값을 구하는 코드에서 사용자의 유형에 따라 arr[i].money.smoney, arr[i].money.cmoney와 같이 표현하게 되면 조금 더 코드의 이해가 수월하다.