

자료구조 실행 파일 보고서

1815060 문정현

1. 스택을 이용한 후위수식 변환 계산기

1) 시작메뉴 화면

```
select(1.infix-postfix 2.exit) ::
```

- 1,2 둘 중 하나의 기능을 선택할 수 있도록 한다

```
select(1.infix-postfix 2.exit) :: 3  
주어진 번호내에서 입력하세요  
select(1.infix-postfix 2.exit) ::
```

-주어진 번호가 아닌 예외 처리

2) 후위수식 변환 계산기능

```
select(1.infix-postfix 2.exit) :: 1  
Enter the expression:: 12*(14-8)
```

-변환계산 기능을 선택한 뒤 중위수식을 입력을 받는다

-후위수식으로 변환하는 과정을 출력하면서 연산자 스택의 상태 또한 같이 출력해준다

-후위수식을 이용하여 식을 계산하는 과정의 스택을 출력해준다

```
postfix:1  
token stack:  
  
postfix:12  
token stack:  
  
postfix:12  
token stack:*(  
  
postfix:12  
token stack:*(  
  
postfix:12 14  
token stack:*(  
  
postfix:12 14  
token stack:-(  
  
postfix:12 14 8  
token stack:-(  
  
postfix:12 14 8 -  
token stack:*(  
  
postfix:12 14 8 - *  
token stack:
```

```
eval stack: 12  
eval stack: 14 12  
eval stack: 8 14 12  
eval stack: 6 12  
eval stack: 72  
answer: 72  
select(1.infix-postfix 2.exit) ::
```

-실행 예시

```
select(1.infix-postfix 2.exit) :: 1
Enter the expression:: 12+3-5*(5+8)
```

```
postfix:1
token stack:
```

```
postfix:12
token stack:
```

```
postfix:12
token stack:+
```

```
postfix:12 3
token stack:+
```

```
postfix:12 3 +
token stack:-
```

```
postfix:12 3 + 5
token stack:-
```

```
postfix:12 3 + 5
token stack:*-
```

```
postfix:12 3 + 5
token stack:(*-
```

```
postfix:12 3 + 5 5
token stack:(*-
```

```
postfix:12 3 + 5 5
token stack:+(*-
```

```
postfix:12 3 + 5 5 8
token stack:+(*-
```

```
postfix:12 3 + 5 5 8 +
token stack:*-
```

```
postfix:12 3 + 5 5 8 + *
token stack:-
```

```
postfix:12 3 + 5 5 8 + * -
token stack:
```

```
eval stack: 12
```

```
eval stack: 3 12
```

```
eval stack: 15
```

```
eval stack: 5 15
```

```
eval stack: 5 5 15
```

```
eval stack: 8 5 5 15
```

```
eval stack: 13 5 15
```

```
eval stack: 65 15
```

```
eval stack: -50
```

```
answer: -50
```

```
select(1.infix-postfix 2.exit) :: 1
Enter the expression:: 12*5-6*(5+6)
```

```
postfix:1
token stack:
```

```
postfix:12
token stack:
```

```
postfix:12
token stack:*
```

```
postfix:12 5
token stack:*
```

```
postfix:12 5 *
token stack:-
```

```
postfix:12 5 * 6
token stack:-
```

```
postfix:12 5 * 6
token stack:*-
```

```
postfix:12 5 * 6
token stack:(*-
```

```
postfix:12 5 * 6 5
token stack:(*-
```

```
postfix:12 5 * 6 5
token stack:+(*-
```

```
postfix:12 5 * 6 5 6
token stack:+(*-
```

```
postfix:12 5 * 6 5 6 +
token stack:*-
```

```
postfix:12 5 * 6 5 6 + *
token stack:-
```

```
postfix:12 5 * 6 5 6 + * -
token stack:
```

```
eval stack: 12
```

```
eval stack: 5 12
```

```
eval stack: 60
```

```
eval stack: 6 60
```

```
eval stack: 5 6 60
```

```
eval stack: 6 5 6 60
```

```
eval stack: 11 6 60
```

```
eval stack: 66 60
```

```
eval stack: -6
```

```
answer: -6
```

```

select(1.infix-postfix 2.exit) :: 1
Enter the expression:: 12+5*8+78

postfix:1
token stack:

postfix:12
token stack:

postfix:12
token stack:+

postfix:12 5
token stack:+

postfix:12 5
token stack:*+

postfix:12 5 8
token stack:*+

postfix:12 5 8 * +
token stack:+

postfix:12 5 8 * + 7
token stack:+

postfix:12 5 8 * + 78
token stack:+

postfix:12 5 8 * + 78 +
token stack:

```

```

eval stack: 12
eval stack: 5 12
eval stack: 8 5 12
eval stack: 40 12
eval stack: 52
eval stack: 78 52
eval stack: 130
answer: 130

```

2. 순환 양방향 연결리스트를 이용한 마블

1) 시작화면

```

1.Throw Dice
2.Show City Info
3.exit

player[0]. choose a menu: _

```

-게임을 시작하기 전 프로그램에서는 미리 게임을 위한 플레이어 2명의 정보를 초기화 시키고 게임을 위한 여행지 정보를 입력을 자동으로 설정해준다

2) 플레이어의 이동

```

player[0]. choose a menu: 1
player[0] dice: 2
Player[0] moves to [일본]
select(1.Buy 2.Pass) : _

```

-주사위는 1~4로 랜덤하게 굴러지고 플레이어가 도착한 위치를 알려준 뒤 플레이어의 행동을 선택할 수 있게 한다

```

player[0] dice: 2
Player[0] moves to [일본]

select(1.Buy 2.Pass) :1
Player[0] buy [일본]

1.Throw Dice
2.Show City Info
3.exit

player[1]. choose a menu: 2

```

NO.	City	(Owner)	Price	User
0	HOME	(-)	0	1
1	중국	(-)	3500	(-)
2	일본	(0)	3000	0
3	미국	(-)	1500	(-)
4	프랑스	(-)	1500	(-)
5	스페인	(-)	2500	(-)
6	이태리	(-)	4000	(-)
7	러시아	(-)	2000	(-)
8	홍콩	(-)	1500	(-)
9	대만	(-)	1000	(-)
10	태국	(-)	4000	(-)

```

Player[0] at [일본] balance (0)
Player[1] at [HOME] balance (3000)

```

-플레이어가 나라를 산 뒤 바뀐 상황을 메뉴에서 2번을 선택하면 바로 확인할 수 있다

```
player[0] dice: 2
Player[0] moves to [프랑스]

select(1.Buy 2.Pass) :1
Your balance is not enough to buy [프랑스]
```

-잔액이 부족하면 구매를 할 수 없고 다음 플레이어의 턴이 시작된다

```
player[1]. choose a menu: 1

player[1] dice: 1
움직이는 방향이 왼쪽으로 변경되었습니다
Player[1] moves to [일본]
```

-만약 플레이어의 주사위가 1일 경우 움직이는 위치를 반대로 변경하고 바뀐 상태를 출력해준다

```
Player[0]에게 통행료[600] 지급
통행료를 받았습니다!!
```

-다른 사람의 소유지에 도착한 경우 통행료를 지급해야 한다

```
월급 입금완료
Player[0] moves to [HOME]

Player[0] arrive at [HOME]
```

-플레이어가 게임 시작 이후 HOME을 지나가면 월급 200을 지급받는다

```
Player[0] moves to [일본]

select(1.Sell 2.Pass) :1
Player[0] sell [일본]
```

-플레이어는 자신의 소유지에 도착하면 매각이 가능하다(기존 가격에 60프로 싼 가격)

3) 게임 정보 보기

```
1.Throw Dice
2.Show City Info
3.exit

player[0]. choose a menu: 2
```

NO.	City	(Owner)	Price	User
0	HOME	(-)	0	0 1
1	중국	(-)	3500	(-)
2	일본	(-)	3000	(-)
3	미국	(-)	1500	(-)
4	프랑스	(-)	1500	(-)
5	스페인	(-)	2500	(-)
6	이태리	(-)	4000	(-)
7	러시아	(-)	2000	(-)
8	홍콩	(-)	1500	(-)
9	대만	(-)	1000	(-)
10	태국	(-)	4000	(-)

```
Player[0] at [HOME] balance (3000)
Player[1] at [HOME] balance (3000)

1.Throw Dice
2.Show City Info
3.exit

player[0]. choose a menu:
```

-각 나라의 가격은 파일 실행할 때마다 랜덤으로 정해지고 메인 메뉴에서 2번을 선택하면 각 나라의 가격, 소유자, 플레이어들의 위치와 잔고를 확인할 수 있다

4) 게임의 종료

-먼저 파산되거나 둘 중 어느 한 명이 여행을 끝내게 되었을 때는 전체 재산(잔고+보유한 여행지의 가격)을 계산하여 우승자를 출력하고 게임이 끝난 뒤 할당 받은 힙 메모리를 해제하고 프로그램은 종료된다

```
Player[1]에게 통행료[300] 지급해야 합니다

Player[0] lose Player[1] win!!

-----
Process exited after 743.5 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .
```