mud_game 구현하기

182659 김성준

서론

- **프로젝트 목적 및 배경**: 2차원 배열, 함수 선언 및 입출력, 반복문이나 조건문을 활용하기 위한 프로그램 구현
- **프로그램 목표** : 간단한 Mud게임 구현하기

요구사항

- **사용자 요구사항**: 유저가 상하좌우로만 이동하며 목적지에 도착하는 게임.
- **기능 계획**

1. 보드판을 초기화 하고 게임 안내와 함께 게임을 시작한다.
2. 사용자의 입력 처리하기
    - 좌표입력이 ‘상’,’하’,’좌’,’우’ 중 하나인 경우
        - 각각의 입력에 대해 유효성 검사
        - 유효한 입력의 경우 좌표 이동
        - 유효하지 않은 입력의 경우 에러메시지 출력 후 사용자 입력단계로 돌아가기
    - 좌표입력이 ‘지도’ 인 경우
        - 함수 호출을 통해 현재 맵의 상태를 출력한다.
    - 좌표입력이 ‘종료’인 경우
        - 게임 종료 안내를 하고 게임을 종료한다
    - 좌표 입력이 잘못된 경우
        - 에러 메시지를 출력하고 사용자 입력 단계로 돌아간다.
3. 목적지 도착 판별 기능: 사용자의 입력을 처리한 후, 목적지 도착 여부를 체크한다.
4. 플레이어 체력 확인 : 1~3의 과정을 수행 후, 플레이어의 체력이 0 이하이면 게임을 종료 한다.

- **함수 계획**

1. 지도와 플레이어의 위치 출력하는 함수:displayMap()
2. 이동 할 좌표의 유효성 체크하는 함수:checkXY()
3. 플레이어의 위치가 목적지 인지 체크하는 함수:checkGoal()
4. 현재 좌표의 상태를 나타내는 함수:checkState()

5. 좌표 이동의 결과를 출력하는 함수: printMovement()

설계 및 구현

기능구현

- 보드판 초기화 및 안내와 함께 게임 시작

```
// 0은 빈 공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지
  int map[mapY][mapX] = { {0, 1, 2, 0, 4},
          {1, 0, 0, 2, 0},
          {0, 0, 0, 0, 0},
          {0, 2, 3, 0, 0},
          {3, 0, 0, 0, 2} };


  // 유저의 위치를 저장할 변수
  int user_x = 0; // 가로 번호
  int user_y = 0; // 세로 번호

  //게임 시작 알림과 함께 게임안내, 현재 위치, 맵의 정보를 출력
  cout << endl << ("mud 게임을 시작합니다. ") << endl;
  cout << endl << ("상하 좌우로 이동할수 있으며 이동시 체력이 1 깎입니다.") << endl;
  cout << endl << ("적을 만나면 체력이 2 깎입니다.") << endl;
  cout << endl << ("보드판을 벗어나거나 비정상 입력인경우 제자리로 돌아옵니다.") << endl;
  cout << endl << ("지도를 입력하면 현재 위치와 맵상황을 보여주고, 종료를 입력시 게임이 종료됩니다.") << endl;
  cout << endl << ("현재의 맵 상황입니다. 건투를 빕니다. ") << endl << endl;


  displayMap(map, user_x, user_y); // 사용자에게 초기 맵상태를1 회 출력합니다.
```

  - 입력
    - user_x: 플레이어 x좌표 값
    - user_y: 플레이어 y좌표 값
    - map[map][mapx] : 보드판 나타내는 2차원배열
      - 0은 빈 공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지를 나타냄
  - 결과
    - 게임 설명을 출력
    - 맵을 초기화 하고 현재 맵의 상태를 출력한다.
  - 설명
    - 보드판을 초기화 하고 사용자의 입력 대기한다.
    - 현재 입출력함수를 통해 사용자의 입력을 받는다.
    - displaymap()을 통해 현재의 맵상태를 출력한다.

- 사용자의 입력 처리하기

```
//사용자의 좌표 이동 처리하기(상,하,좌,우 중 하나인경우 )
    if (user_input == "상" || user_input == "하" || user_input == "좌" || user_input == "우") {
```

```
      //사용자의  좌표이동  처리하기
      bool inMap;//좌표 유효성 검사 결과를 저장하는 변수
      int movementX = 0, movementY = 0; // x좌표,y좌표의 이동을 임시로 저장하는 변수.사용자 입력이 들어올때마다 초기화 된다.

      //사용자의 좌표의 입력을 통해, 각각상황에 맞게  checkXY 함수 호출 및 movemntX,movementY변수 초기화
      if (user_input == "상") {
        inMap = checkXY(user_x, mapX, user_y - 1, mapY);
        movementY--;
      }
      else if (user_input == "하") {
        inMap = checkXY(user_x, mapX, user_y + 1, mapY);
        movementY++;
      }
      else if (user_input == "좌") {
        inMap = checkXY(user_x - 1, mapX, user_y, mapY); //좌표의 유효성 검사 결과를 inamp 변수에 저장
        movementX--;
      }
      else { // 사용자의 입력이"하 " 인경우
        inMap = checkXY(user_x + 1, mapX, user_y, mapY);
        movementX++;
      }

      if (inMap == true) { //사용자의 좌표이동이 유효한 경우에만 아래 과정을 수행한다.

        //사용자의 좌표 이동
        user_x += movementX;
        user_y += movementY;
        printMovement(movementX,movementY);
        user_hp--; //사용자 이동시 체력 1씩 감소
        checkState(map, user_x, user_y);
        displayMap(map, user_x, user_y);
      }
      else {//사용자의 좌표이동이 유효하지 않은경우 출력
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
      }
    }
```

  - 입력
    - inMap : 좌표 유효성 검사 결과를 저장하는 변수
    - movementX ,movementY : x좌표 y좌표의 이동을 임시로 저장하는 변수
  - 결과
    - 입력이 유효한 경우 각 입력에 맞게 좌표를 이동시킨다.
    - 입력이 맵을 벗어난경우 에러메시지를 출력한다
    - 좌표 이동이 일어난 경우 사용자 체력,chekState( ), displayMap( ) 를 통해 현재 좌표의 상태, 맵의 상태를 업데이트 한다.
  - 설명
    - 상/하/좌/우의 이동, 지도,종료, 등에서 사용자의 선택 입력을 받는다.
    - 선택 입력이 지도인 경우 지도를 출력하고, 종료인 경우 게임을 종료한다.
    - 선택 입력이 상/하/좌/우 이동인 경우 각 이동 입력에 따라 checkXY() 함수를 호출하여 유효성 검사 결과를 inMap 변수에 저장한다.
    - 각 입력에 따라 movement를 정의 하고 , inMap이 true 인 경우 printMovement를 호출하여 걸과를 출력한다.
- 목적지 도착 판별하기

```
// 목적지에 도달했는지 체크
    bool finish = checkGoal(map, user_x, user_y);
    if (finish == true) { // 목적지에 도달한경우 출력
      cout << "목적지에 도착했습니다! 축하합니다!" << endl;
      cout << "게임을 종료합니다." << endl;
      break;
    }
```

- 입력
  - bool finish: 목적지 도착여부 저장하는 변수
- 결과
  - finish: true 인경우 도착메시지를 출력하고 게임을 종료한다.
- 설명
  - checkGoal()함수를 호출하여 결과를 finish 에 저장한다..
  - 조건문을 통해 결과를 비교하고 결과를 출력하거나 다음 반복을 실행한다.

- 플레이어 체력 확인

```
//hp가 0 이하가 되면 "실패"를 입력하고 종료
    if (user_hp <= 0) {
      cout << "HP가 0 이하가 되었습니다. 실패 했습니다" << endl;
      cout << "게임을 종료합니다." << endl;
      break;
    }
```

- 입력
  - user_hp : 사용자의 체력을 저장하는 변수
- 결과
- 플레이어의 체력이 0 이하임을 (hp≤0) 임을 출력하고 게임 종료 또는 게임 재개

함수 구현

- 지도와 사용자 위치 출력하는 함수

```
// 지도와 사용자 위치 출력하는 함수
void displayMap(int map[][mapX], int user_x, int user_y) {
  for (int i = 0; i < mapY; i++) {
    //각 행이 바뀔때마다 보드의 테두리를 그린다.
    cout << " ---------------------------------- " << endl;
    cout << "|";
    for (int j = 0; j < mapX; j++) {
      if (i == user_y && j == user_x) {//해당좌표에 사용자가 있는경우
        cout << " USER |"; // 양 옆 1칸 공백
      }
      else { //그외 동일하게 출력
        int posState = map[i][j];
        switch (posState) {
        case 0:
          cout << "      |"; // 6칸 공백
          break;
        case 1:
          cout << "아이템|";
          break;
```

```
        case 2:
          cout << "  적  |"; // 양 옆 2칸 공백
          break;
        case 3:
          cout << " 포션 |"; // 양 옆 1칸 공백
          break;
        case 4:
          cout << "목적지|";
          break;
        }
      }
    }
    cout << endl;

  }
  cout << " ---------------------------------- " << endl;
}
```

- 입력
  - map[mapY][mapX] : 맵을 나타내는 2차원 배열
  - user_x,user_y: 사용자의 위치를 나타내는 변수
  - postate : 맵의 구조물을 표현하는 변수
- 설명
  - postate 변수에 각 맵의 구조물의 위치를 받는다.
  - 각 행을 개행 할때마다 보드의 윗줄과 제일 왼쪽줄을 추가한다.
  - 각 칸마다 switch 조건문을 활용하여 보드판을 채운다.
- 이동하려는 위치 체크하는 함수

```
// 이동하려는 곳이 유효한 좌표인지 체크하는 함수
bool checkXY(int user_x, int mapX, int user_y, int mapY) {
  bool checkFlag = false;
  if (user_x >= 0 && user_x < mapX && user_y >= 0 && user_y < mapY) {
    checkFlag = true;
  }
  return checkFlag;
}
```

- 입력:
  - user_x,user_y ,mapX,mapY(설명 생략)
  - chekFlag : 유효성 검사 결과 저장
- 설명
  - user_x, user_y 가 각각 0 이상이고 map안에 들어가는지 확인하고 그 결과를 반환한다.
- 현재 좌표의 상태를 나타내는 함수

```
//현재 좌표의 상태를 나타내는 함수
void checkState(int map[][mapX], int user_x, int user_y) {
  int check = map[user_y][user_x];

  switch (check) {
```

```
  case 1:
    cout << "아이템이 있습니다" << endl;

    break;
  case 2:
    cout << "  적이 있습니다. HP가 2가 줄어듭니다." << endl;
    user_hp = user_hp - 2;
    break;
  case 3:
    cout << " 포션이 있습니다. HP가 2 늘어납니다." << endl;
    user_hp = user_hp + 2;
    break;
  }

}
```

- 입력
  - check: 현재 좌표의 상태 나타내는 변수
  - user_hp: 플레이어의 체력
- 설명
  - 현재 좌표의 값을 check변수에 저장하고 swich 조건문 비교한다.
  - 각 조건에 따라 결과를 출력하고 플레이어hp를 조정한다.
- 좌표 이동 결과를 출력하는 함수

```
//좌표이동결과를 출력
void printMovement(int movementX, int movementY) {

  switch (10 * movementX + movementY) {
  case(-1): //상으로 한칸
    cout << "위로 한 칸 올라갑니다." << endl;
    break;
  case(1):///하로 한칸
    cout << "밑으로 한 칸 내려갑니다." << endl;
    break;
  case(-10): //좌로 한칸
    cout << "왼쪽으로 이동합니다." << endl;
    break;
  case(10): // 우로 한칸
    cout << "오른쪽으로 이동합니다." << endl;
    break;
  }
}
```

- 입력:
  - movementX, movementY: 사용자의 이동입력 저장
- 설명:
  - 사용자 이동 입력을 받아서 switch 조건 비교를 한다.
  - x좌표,y좌표의 이동을 동시에 확인하기 위해 나눗셈 연산을 실시하였다.

테스트

- 기능 별 테스트 결과: (요구사항 별 스크린샷)

  1. 보드판을 초기화 하고 게임 안내와 함께 게임을 시작한다.

```
mud 게임을 시작합니다.
상하 좌우로 이동할수 있으며 이동시 체력이 1 깎입니다.
적을 만나면 체력이 2 깎입니다.
보드판을 벗어나거나 비정상 입력인경우 제자리로 돌아옵니다.
지도를 입력하면 현재 위치와 맵상황을 보여주고, 종료를 입력시 게임이 종료됩니다.
현재의 맵 상황입니다. 건투를 빕니다.
 ------------------------------
| USER |아이템|  적  |     |목적지|
 ------------------------------
|아이템|     |     |  적  |     |
 ------------------------------
|     |     |     |     |     |
 ------------------------------
|     |  적  | 포션 |     |     |
 ------------------------------
| 포션 |     |     |     |  적  |
 ------------------------------

현재의 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료):
```

2. 사용자의 입력 처리하기
    - 좌표입력이 '상','하','좌','우' 중 하나인 경우

```
 ---------------------------------

현재의 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
맵을 벗어났습니다. 다시 돌아갑니다.

현재의 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
밑으로 한 칸 내려갑니다.
아이템이 있습니다
 ---------------------------------
|      |아이템|  적  |      |목적지|
 ---------------------------------
| USER |      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------

현재의 HP: 19 명령어를 입력하세요 (상,하,좌,우,지도,종료): 좌
맵을 벗어났습니다. 다시 돌아갑니다.

현재의 HP: 19 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
 ---------------------------------
|      |아이템|  적  |      |목적지|
 ---------------------------------
|아이템| USER |      |  적  |      |
 ---------------------------------
```

        - 각각의 입력에 대해 유효성 검사
        - 유효한 입력의 경우 좌표 이동
        - 유효하지 않은 입력의 경우 에러메시지 출력 후 사용자 입력단계로 돌아가기
    - 좌표입력이 '지도' 인 경우

```
현재의 HP: 18 명령어를 입력하세요 (상,하,좌,우,지도,종료): 지도
 ---------------------------------
|      |아이템|  적   |      |목적지|
 ---------------------------------
|아이템| USER |       |  적  |      |
 ---------------------------------
|      |      |       |      |      |
 ---------------------------------
|      |  적  | 포션  |      |      |
 ---------------------------------
| 포션 |      |       |      |  적  |
 ---------------------------------
```

- 함수 호출을 통해 현재 맵의 상태를 출력한다.

- 좌표입력이 '종료'인 경우

```
현재의 HP: 18 명령어를 입력하세요 (상,하,좌,우,지도,종료): 지도
 ---------------------------------
|      |아이템|  적   |      |목적지|
 ---------------------------------
|아이템| USER |       |  적  |      |
 ---------------------------------
|      |      |       |      |      |
 ---------------------------------
|      |  적  | 포션  |      |      |
 ---------------------------------
| 포션 |      |       |      |  적  |
 ---------------------------------

현재의 HP: 18 명령어를 입력하세요 (상,하,좌,우,지도,종료): 종료
종료합니다.
C:₩Users₩182659₩source₩repos₩Project6₩Debug₩Project6.exe(프로세스 5676개)이(가) 종료
되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이
중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

- 게임 종료 안내를 하고 게임을 종료한다

- 좌표 입력이 잘못된 경우

```
현재의 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료): GK
잘못된 입력입니다.
```

- 에러 메시지를 출력하고 사용자 입력 단계로 돌아간다.

3. 목적지 도착 판별 기능: 사용자의 입력을 처리한 후, 목적지 도착 여부를 체크한다.

```
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
```

1. 플레이어 체력 확인 : 1~3의 과정을 수행 후, 플레이어의 체력이 0 이하이면 게임을 종료 한다.

```
현재의 HP: 1 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
  적이 있습니다. HP가 2가 줄어듭니다.
 ---------------------------------
|      |아이템| USER |      |목적지|
 ---------------------------------
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------
HP가 0 이하가 되었습니다. 실패 했습니다
게임을 종료합니다.

C:₩Users₩182659₩source₩repos₩Project6₩Debug₩Project6.exe(프로세스 22376개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

- 최종 테스트 스크린샷: (프로그램 전체 동작 스크린샷)

```
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------

현재의 HP: 11 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
 ---------------------------------
|      |아이템|  적  |      | USER |
 ---------------------------------
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.

C:₩Users₩182659₩source₩repos₩Project6₩Debug₩Project6.exe(프로세스 24548개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

결과 및 결론

최근에 배웠던 함수 선언 및 호출을 이용하여 반복되는 코드들을 최대한 함수화 하려고 고민을 했습니다. 기존에 제출했던 코드는 조건문,반복문이 너무 많아서 직관적으로 구조를 파악하기 힘들었는데 함수화 하고 보니 오류를 찾기도 쉽고 보기에 용이하다는 생각을 했습니다. 기존에 제출했던 코드의 보드판 출력을 개선했습니다. 함수화를 하려다 보니 이름이 비슷한 변수나 함수들이 많아서 헷갈리지 않도록 함수나 변수의 이름을 직관적이고, 간단하게 지어야겠다고 생각했습니다.