VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Kỹ Thuật Lập Trình (Cơ bản và nâng cao C++)

KTLT2 - HK242

TASK 6 ÔN TẬP TỪ ĐẦU ĐẾN CON TRỎ

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 2/2025

Mục lục

1 Kiến Thức ôn lại KTLT1

- TASK 0 Chạy SRC VÀ DEBUG
- TASK 1 Kiến thức Nhập Môn Điện Toán
- TASK 2 ARRAY, STRING, STRUCT, FILE, ENUM
- TASK 3 FUNCTION
- TASK 4 POINT

2 Cách Chạy Code

https://discord.com/channels/1343546610897915995/1343872430266912819

3 Xây dựng Configuration trong BTL2

3.1 Mô tả Configuration.h

1. `enum InfantryType` (Loại lính bộ binh) Định nghĩa các loại lính bộ binh trong mô phỏng chiến trường:
   - `SNIPER`: Lính bắn tỉa.
   - `ANTIAIRCRAFTSQUAD`: Đội phòng không.
   - `MORTARSQUAD`: Đội súng cối.
   - `ENGINEER`: Công binh.
   - `SPECIALFORCES`: Lực lượng đặc biệt.
   - `REGULARINFANTRY`: Bộ binh thường.
2. `enum TerrainType` (Loại địa hình) Định nghĩa các loại địa hình xuất hiện trên bản đồ chiến trường:
   - `ROAD`: Đường mòn.
   - `FOREST`: Rừng núi.
   - `RIVER`: Sông suối.
   - `FORTIFICATION`: Công sự/chiến hào.
   - `URBAN`: Khu dân cư.
   - `SPECIAL_ZONE`: Khu vực đặc biệt (chùa, bệnh viện, nhà thờ,...).
3. `struct UNIT_NAME` (Thông tin về một đơn vị lính) Cấu trúc lưu trữ thông tin của một đơn vị quân đội trên chiến trường, bao gồm:
   - `quantity`: Số lượng binh lính trong đơn vị.
   - `weight`: Trọng lượng (có thể đại diện cho trọng tải quân trang hoặc phương tiện).
   - `position_x`: Vị trí hoành độ (x) trên bản đồ.
   - `position_y`: Vị trí tung độ (y) trên bản đồ.
   - `armyBelong`: Đội quân thuộc phe nào (`true`: phe ta, `false`: phe địch).
   - `type`: Loại lính bộ binh (`InfantryType`).

3.2 Các Hàm thực thi

3.2.1 Hàm print_battlefield

Chức năng: Hàm này có nhiệm vụ in ra ma trận chiến trường, trong đó mỗi ô trên bản đồ được biểu diễn bằng 3 ký tự đầu tiên của tên loại địa hình (`TerrainType`).

Khai báo hàm:

```
string print_battlefield(const TerrainType **battlefield,
                         const int &NUM_ROWS,
                         const int &NUM_COLS);
```

Tham số đầu vào:

- `battlefield`: Con trỏ cấp hai tới mảng chứa thông tin địa hình của chiến trường.
- `NUM_ROWS`: Số hàng trong ma trận.
- `NUM_COLS`: Số cột trong ma trận.

Cách hoạt động:

1. Duyệt qua từng ô trong ma trận kích thước `NUM_ROWS` x `NUM_COLS`.
2. Lấy giá trị trong ma trận `battlefield[i][j]` và ánh xạ nó sang tên loại địa hình tương ứng.
3. Chỉ lấy 3 ký tự đầu của tên địa hình để hiển thị.
4. Ghép tất cả các giá trị vào một chuỗi theo định dạng bảng.

Ví dụ:

Dữ liệu đầu vào (`battlefield`):

$$\begin{bmatrix} \texttt{ROAD} & \texttt{FOREST} & \texttt{RIVER} \\ \texttt{FORTIFICATION} & \texttt{URBAN} & \texttt{SPECIAL\_ZONE} \end{bmatrix}$$

Kết quả trả về (Chuỗi in ra màn hình):

```
ROA FOR RIV
FOR URB SPE
```

Giá trị trả về: Hàm trả về một chuỗi `string` chứa nội dung ma trận chiến trường, mỗi ô hiển thị 3 ký tự đầu của loại địa hình tương ứng. (không có newline ở hàng cuối cùng)

3.2.2 Hàm `print_UNIT_NAME`

Chức năng: Hàm này in ra danh sách các đơn vị quân theo định dạng:

```
[
    TANK(quantity, weight, (position_x, position_y), armyBelong),
    REGULARINFANTRY(quantity, weight, (position_x, position_y), armyBelong)
]
```

Khai báo hàm:

```
string print_UNIT_NAME(const vector<UNIT_NAME> &units);
```

Tham số đầu vào:

- `units`: Danh sách các đơn vị quân (`UNIT_NAME`).

Cách hoạt động:

1. Tạo một chuỗi kết quả theo định dạng danh sách.
2. Duyệt qua từng đơn vị quân trong danh sách `units`.
3. Lấy tên `InfantryType` của đơn vị và đưa vào định dạng:

   `InfantryType(quantity, weight, (position_x, position_y), armyBelong)`
4. Thêm ký tự xuống dòng và thụt đầu dòng *Tab* trước mỗi đơn vị.
5. Kết thúc danh sách bằng dấu ']'.

Ví dụ:

Dữ liệu đầu vào:

```
vector<UNIT_NAME> units = {
    {5, 2, 1, 2, false, TANK},
    {5, 2, 3, 2, true, TANK},
    {5, 2, 1, 1, false, REGULARINFANTRY},
    {5, 2, 3, 3, true, REGULARINFANTRY}
};
```

Kết quả trả về:

```
[
    TANK(5,2,(1,2),0),
    TANK(5,2,(3,2),1),
    REGULARINFANTRY(5,2,(1,1),0),
    REGULARINFANTRY(5,2,(3,3),1)
]
```

Giá trị trả về: Hàm trả về một chuỗi `string` chứa danh sách các đơn vị quân theo định dạng yêu cầu. (không có newline ở hàng cuối cùng)

3.2.3 Hàm `clearBattlefield`

Chức năng: Hàm này đặt lại giá trị của toàn bộ ma trận chiến trường (`battlefield`) về loại địa hình mặc định (`ROAD`) và đồng thời giải phóng bộ nhớ đã cấp phát động cho ma trận.

Khai báo hàm:

```
void clearBattlefield(TerrainType **battlefield, const int &NUM_ROWS, const int
 ↪  &NUM_COLS);
```

Tham số đầu vào:

- `battlefield`: Con trỏ đến ma trận chiến trường, nơi mỗi ô chứa một giá trị thuộc kiểu `TerrainType`.
- `NUM_ROWS`: Số hàng của ma trận chiến trường.
- `NUM_COLS`: Số cột của ma trận chiến trường.

Cách hoạt động:

1. Duyệt qua từng ô trong ma trận chiến trường (`battlefield`) và đặt giá trị mặc định là `ROAD`.
2. Giải phóng bộ nhớ đã cấp phát động cho từng hàng của ma trận.
3. Giải phóng con trỏ cấp cao nhất của ma trận.

4. Đặt con trỏ `battlefield` về `nullptr` để tránh lỗi truy cập bộ nhớ.

Ví dụ:

Trước khi gọi hàm:

```
battlefield =
[
    {FOREST, RIVER, ROAD},
    {URBAN,  ROAD,  FORTIFICATION},
    {SPECIAL_ZONE, FOREST, ROAD}
]
```

Sau khi gọi hàm (bộ nhớ đã được giải phóng, con trỏ bị vô hiệu hóa):

```
battlefield = nullptr;
```

Giá trị trả về: Hàm không có giá trị trả về, nhưng thay đổi trực tiếp nội dung của ma trận `battlefield` và giải phóng bộ nhớ động.

Lưu ý:

- Hàm này đảm bảo rằng không có rò rỉ bộ nhớ do bộ nhớ động của ma trận được giải phóng đúng cách.
- Sau khi gọi hàm, con trỏ `battlefield` sẽ trở thành `nullptr`, vì vậy không nên sử dụng lại nếu chưa được cấp phát lại.

3.2.4 Hàm `processFile`

Chức năng: Hàm này đọc dữ liệu từ tập tin cấu hình đầu vào, phân tích và lưu thông tin vào các biến tương ứng, bao gồm kích thước bản đồ, bố trí địa hình, danh sách đơn vị quân sự và mã sự kiện.

Khai báo hàm:

```
void processFile(const string &filename,
                 TerrainType ** &battlefield,
                 int &NUM_ROWS,
                 int &NUM_COLS,
                 vector<UNIT_NAME> &units,
                 int &EVENT_CODE);
```

Tham số đầu vào:

- `filename`: Tên tập tin chứa dữ liệu cấu hình.
- `battlefield`: Ma trận chiến trường, nơi các ô có thể chứa thông tin địa hình.
- `NUM_ROWS`: Biến lưu số hàng của bản đồ trận địa.
- `NUM_COLS`: Biến lưu số cột của bản đồ trận địa.
- `units`: Danh sách các đơn vị quân sự có mặt trên chiến trường.
- `EVENT_CODE`: Mã sự kiện xác định tình trạng chiến sự.

Định dạng tập tin đầu vào: Tập tin cấu hình có thể chứa các dòng theo thứ tự bất kỳ với các định dạng sau:

1. `NUM_ROWS=<nr>`
2. `NUM_COLS=<nc>`
3. `ARRAY_FOREST=[<pos1>,<pos2>,..]`
4. `ARRAY_RIVER=[<pos1>,<pos2>,..]`

5. ARRAY_FORTIFICATION=[<pos1>,<pos2>,..]
6. ARRAY_URBAN=[<pos1>,<pos2>,..]
7. ARRAY_SPECIAL_ZONE=[<pos1>,<pos2>,..]
8. UNIT_LIST=[<unit1>,<unit2>,...]
9. EVENT_CODE=<ec>

Trong đó:

- <nr> là số hàng của bản đồ trận địa, ví dụ:
- <nc> là số cột của bản đồ trận địa, ví dụ:
- [<pos1>,<pos2>,..] là vị trí của các yếu tố địa hình trên trận địa. Ví dụ: là vị trí các yếu tố rừng rậm (FOREST) trong bản đồ trận địa.
- [<unit1>,<unit2>,...] là danh sách các đơn vị quân sự của cả hai quân đội. Mỗi đơn vị (UNIT_NAME) có định dạng, với armyBelong=0 tương ứng với Quân Giải phóng, armyBelong=1 tương ứng với Quân đội chính quyền Sài Gòn.
- <ec> là mã sự kiện để xác định quân đội nào đang trong thế tấn công hay phòng thủ ở trận địa hiện tại.

Ví dụ:

```
NUM_ROWS=10
NUM_COLS=8
ARRAY_FOREST=[(1,2),(3,5)]
ARRAY_RIVER=[(0,0),(0,4)]
ARRAY_FORTIFICATION=[(6,6)]
ARRAY_URBAN=[(2,0)]
ARRAY_SPECIAL_ZONE=[(9,7)]
UNIT_LIST=[
    TANK(5,2,(1,2),0),
    TANK(5,2,(3,2),1),
    REGULARINFANTRY(5,2,(1,1),0),
    REGULARINFANTRY(5,2,(3,3),1)
]
EVENT_CODE = 23
```