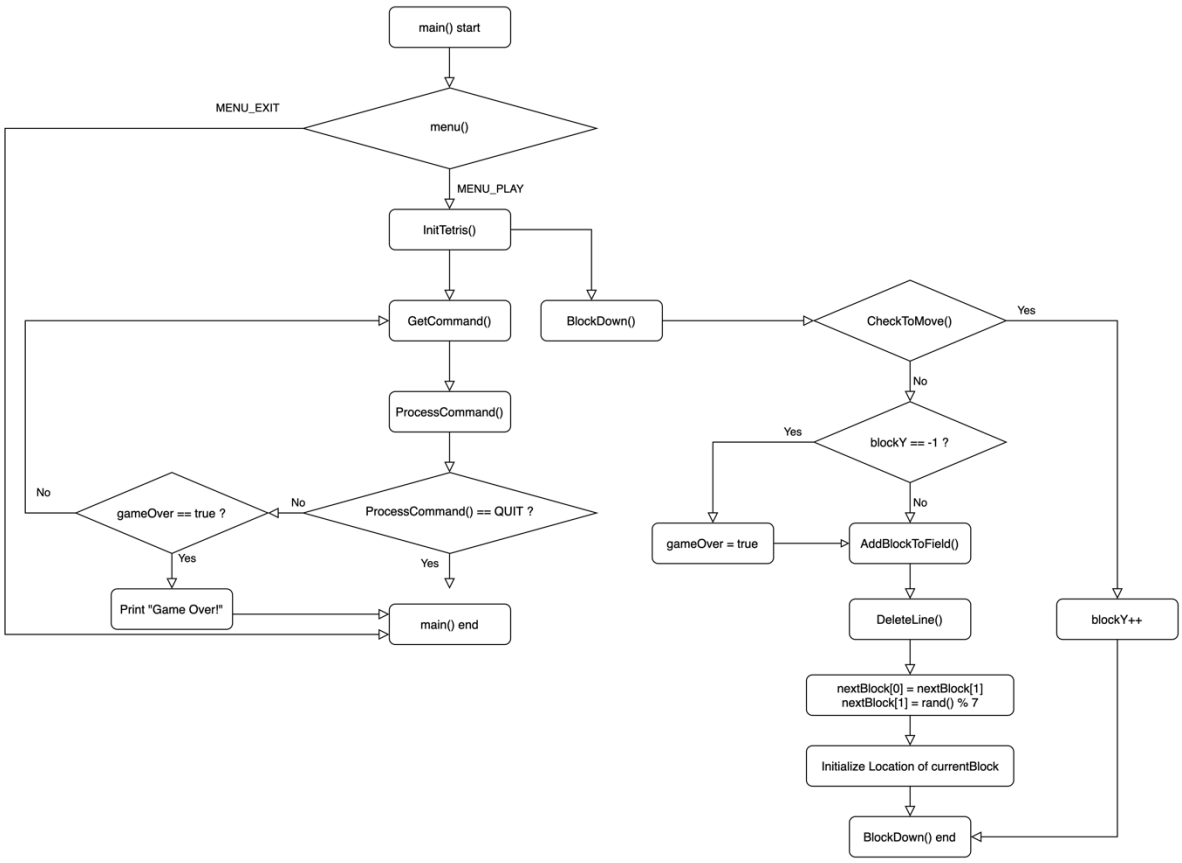
1. 테트리스 게임의 flow chart 및 각 함수의 기능

(1) flow chart



(2) 구성 함수의 기능

void InitTetris()

변수와 자료구조를 초기화하고, 테트리스 초기 화면을 그린다.

void DrawOutline()

테트리스 field의 테두리, 다음 블록을 보여주는 부분의 테두리, 점수를 보여주는 부분의 테두리를 그린다.

int GetCommand()

사용자의 입력(←, ↑, →, ↓, space, Q)을 받고 구분하여 입력받은 명령을 리턴한다.

int ProcessCommand(int command)

입력받은 명령을 전달받아 어떤 명령인지에 따라 다른 수행을 한다. 명령이 QUIT이면 0, 아니면 1을 리턴한다.

void BlockDown(int sig)

현재 블록을 아래로 내린다. 만약 더 이상 내릴 수 없다면 블록을 필드에 쌓고, 빈 칸 없는 줄이 있는지 확인해서 삭제하고, 점수를 계산한다. 또한 필드에 블록을 놓을 공간이 없다면 게임을 종료한다.

int CheckToMove(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX)

블록이 명령에 따라 이동할 수 있는지 없는지 판단해 있다면 1, 없다면 0을 리턴한다.

void DrawChange(char f[HEIGHT][WIDTH],int command,int currentBlock,int blockRotate, int blockY, int blockX)

명령에 따라 이동한/회전한 블록을 다시 그린다. (이전 블록을 지우고, 새 블록을 그린다.)

void DrawField()

테트리스 필드를 그린다.

void AddBlockToField(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX)

해당 좌표에 맞게 필드에 현재 블록을 쌓는다.

int DeleteLine(char f[HEIGHT][WIDTH])

빈 칸이 없는 줄을 찾아서 지우고, 지워진 줄 개수에 따라 점수를 계산해 리턴한다.

void gotoyx(int y, int x)

커서의 위치를 해당 좌표로 이동시킨다.

void DrawNextBlock(int *nextBlock)

화면 오른쪽 상단에 다음 블록을 그려준다.

void PrintScore(int score)

화면 오른쪽 하단에 점수를 보여준다.

void DrawBox(int y,int x, int height, int width)

해당 좌표에 해당 크기의 박스를 그린다.

void DrawBlock(int y, int x, int blockID,int blockRotate,char tile)

해당 좌표에 해당 모양 및 회전수의 블록을 그린다.

void DrawShadow(int y, int x, int blockID,int blockRotate)

블록이 떨어지면 위치할 장소를 그림자로 그린다.

void Play()

테트리스 게임을 시작한다.

char menu()

메뉴를 출력한다.

void createRankList()

랭킹 기록 파일로부터 정보를 읽어 랭킹 목록을 구성한다.

void rank()

랭킹 기록을 화면에 보여준다.

void writeRankFile()

랭킹 기록 파일을 생성한다.

void newRank(int score)

새 랭킹 정보를 추가한다.

int recommend(RecNode *root)

블록을 쌓을 추천 위치를 구한다.

void recommendedPlay()

추천 기능에 따라 블록을 위치시키면서 진행되는 게임을 시작한다.

2. 실습 시간에 구현할 함수 pseudo code

```
int CheckToMove(char f[HEIGHT][WIDTH], int currentBlock, int blockRotate,
                int blockY, int blockX)
{
       for (i = 0 to BLOCK_HEIGHT-1)
              for (j = 0 to BLOCK_WIDTH-1)
                     if (block[currentBlock][blockRotate][i][j] == 1)
                            x = blockX + j
                            y = blockY + i
                            if (!(0≤x<WIDTH && 0≤y<HEIGHT))
                                   return 0
                            if (f[y][x] == 1)
                                   return 0
       return 1
}

void DrawChange(char field [HEIGHT][WIDTH],int command, int currentBlock,
                int blockRotate, int blockY, int blockX){
       if (command == KEY_UP) pre_blockRotate = (blockRotate+3)%4
       else if (command == KEY_DOWN) pre_blockY -= 1
       else if (command == KEY_LEFT) pre_blockX += 1
       else if (command == KEY_RIGHT) pre_blockX -= 1

       for (i = 0 to BLOCK_HEIGHT-1)
              for (j = 0 to BLOCK_WIDTH-1)
                     if (block[currentBlock][pre_blockRotate][i][j] == 1
                            && i + pre_blockY >= 0)
                            move cursor to (i+pre_blockY+1, j+prev_blockX+1)
                            print “.”

       DrawBlock(blockY, blockX, currentBlock, blockRotate, ‘ ‘)
}
```

```
void BlockDown(int sig)
{
	if (block can go down)
		blockY++
		DrawChange(field, KEY_DOWN, nextBlock[0], blockRotate, blockY,
				blockX)
	else
		if (blockY == -1) gameOver = 1
		AddBlockToField(field, nextBlock[0], blockRotate,
					blockY, blockX)
		score += DeleteLine(field)
		nextBlock[0] = nextBlock[1]
		nextBlock[1] = rand() % 7
		blockY = -1
		blockX = WIDTH/2 – 2
		blockRotate = 0
		DrawNextBlock(nextBlock)
		PrintScore(score)
		DrawField()
	timed_out = 0
}

void AddBlockToField(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate,
		int blockY, int blockX)
{
	for (i = 0 to BLOCK_HEIGHT-1)
		for (j = 0 to BLOCK_WIDTH-1)
			if (block[currentBlock][blockRotate][i][j] == 1)
				y = blockY + i
				x = blockX + j
				f[y][x] = 1
}
```

```
int DeleteLine(char f[HEIGHT][WIDTH])
{
	count = 0
	for (i = 0 to HEIGHT-1)
		for (j = 0 to WIDTH-1)
			if (!f[i][j])
				Set filled_flag to False and escape this loop
			else
				Set filled_flag to True
		if (filled_flag)
			count++
			for (y = i downto 0)
				for (x = 0 to WIDTH-1)
					f[y][x] = f[y-1][x]
			for (x = 0 to WIDTH-1)
				f[0][x] = 0
			i--;
	return count*count*100
}
```