Ncurses를 이용한

SnakeGame

1조 _ 소프트웨어학과 20171703 정태원

정보보안암호수학과 20182251 최지원

INDEX

□ 프로젝트 주제

“ Ncurses library를 이용해서
Snake Game 만들기 ”

□ 0단계

목표 | SnakeGame를 객체로 만들기

SnakeGame

Attribute

| Map 관련 변수들
| Snake 관련 변수들
| Growth item 관련 변수들
| Poison item 관련 변수들
| Gate 관련 변수들
| ScoreBoard 관련 변수들
| 추가한 규칙 관련 변수들

Method

| Map 관련 함수들
| Snake 관련 함수들
| Growth item 관련 함수들
| Poison item 관련 함수들
| Gate 관련 함수들
| ScoreBoard 관련 함수들
| 추가한 규칙 관련 함수들

□ 0단계

02 단계별 수행 과정

목표 | Snake를 객체로 만들기

※ 아래 코드는 최종적으로 완성된 SnakeGame 객체

```
struct point{
  int x;
  int y;
};

class SnakeGame
{
private:
  ///* Attribute *///
  /* Map parameter */
  int row=0, col=0;

  /* Snake parameter */
  queue<pair<int,int>> xy; // queue for storing snake location
  point head; // x, y of head
  point body; // x, y of body
  point tail; // x, y of tail
  int max_length; // maximum length of snake
  int length=3; // length of snake (snake length is 3 at the start)
  char direction; // direction of snake

  /* ScoreBoard parameter */
  int score; // score of game
  int tick; // checking time delay
  int height, width;
  int vertical , horizontal; // length of terminal
  int Randheight, Randwidth; // location of snake head
  int ch; // keyboard
  point start; // x,y of start
  int count1=0,count2=0,count3=0,count4=0,countCobra=0; // time count for
  int num_gate=0; // number of gate snake passed
``
``  /* Wall parameter */
  int wall[21][42]={0,}; // wall->1, not wall->0

  /* Gate parameter */
  point gate1; // x,y of gate1
  point gate2; // x,y of gate2
  point in; // x,y of entrance
  point out; // x,y of exit

  /* Cobra parameter */
  queue<pair<int,int>> cobra; // queue for storing snake location
  point cbody; // x, y of cobra body
  char cdirection; // direction of cobra

  /* Growth item parameter */
  point growth; // x,y of growth item
```

```
  int num_poison=0; // number of poison item

  /* Hole parameter */
  point hole; // x,y of hole

  /* Mission parameter */
  int mission_length; // mission length of snake
  int mission_growth; // mission number of growth item
  int mission_poison; // mission number of poison item
  int mission_gate; // mission number of gate


  ///* Method *///
  void StartWindow(); // initalize window

  /* Snake function */
  void LocateSnake(); // put snake on the map
  void WrongDirection(); // if snake moves to wrong direction, game over
  void BumpedintoBody(); // check whether snake head bumped into body or
  void SizeofSnake(); // check whether snake length is shorter than 3
  int ControlSnake(); // get input + movement by tick
  bool IsBody(int,int); // check whether snake head bumped into body

  /* Map function */
  void MakeMap1(); // make map1
  void MakeMap2(); // make map2
  void MakeMap3(); // make map3
  void MakeMap4(); // make map4

  /* Wall function */
  void BumpedintoWall(); // check whether snake bumped into wall or not

  /* Growth item function */
  void RandGrowthItem(); // put growth item(5) randomly on the map
  bool EatGrowthItem(); // check whether snake eat growth item or not

  /* Poison item function */
  void RandPoisonItem(); // put poison item(6) randomly on the map
  bool EatPoisonItem(); // check whether snake eat poison item or not

  /* Hole function */
  void RandHole(); // put hole(@) randomly on the map
  void BumpedintoHole(); // check whether snake bumped into hole or not
  bool IsHole(); // check whether snake fall into a hole

  /* Cobra function */
  void BumpedintoCobra(); // check whether snake bumped into cobra or not
  void DangerousCobra(); // watch out for cobra!!!
```

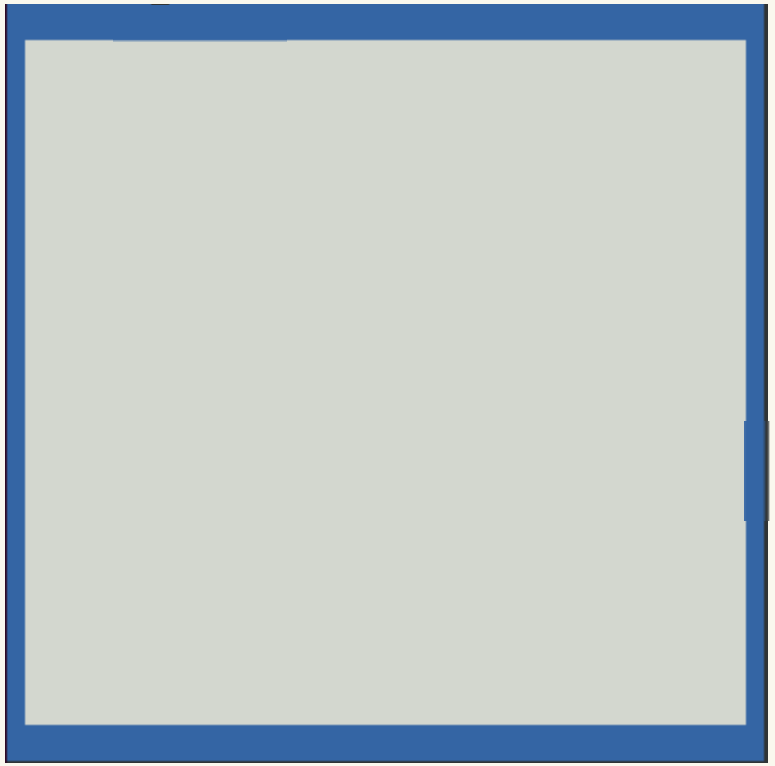
```

  /* Wall function */
  void BumpedintoWall(); // check whether snake bumped into wall or not

  /* Growth item function */
  void RandGrowthItem(); // put growth item(5) randomly on the map
  bool EatGrowthItem(); // check whether snake eat growth item or not

  /* Poison item function */
  void RandPoisonItem(); // put poison item(6) randomly on the map
  bool EatPoisonItem(); // check whether snake eat poison item or not

  /* Hole function */
  void RandHole(); // put hole(@) randomly on the map
  void BumpedintoHole(); // check whether snake bumped into hole or not
  bool IsHole(); // check whether snake fall into a hole

  /* Cobra function */
  void BumpedintoCobra(); // check whether snake bumped into cobra or not
  void DangerousCobra(); // watch out for cobra!!!
  void LocateCobra(); // locate cobra first
  bool IsCobra(); // check whether snake bumped into cobra
  bool DestroyedbyCobra(int,int); // check whether cobra destroy item(gro

  /* Gate function */
  void RandGate(); // put gate(@) randomly on the map
  void ControlGate(); // rule of using gate
  bool IsGate(); // check whether snake head bumped into gate
  bool IsEdgeGate(); // check whether gate is at edge or not

  /* ScoreBoard function */
  void ScoreBoard(); // scoreboard

  /* Mission function */
  bool CheckMission(); // check whether player complete mission or not


public:
  SnakeGame(); // constructor
  ~SnakeGame(); // destructor
  void Start(int); // start new snake game
  bool death = false; // check death
  //bool GameStart = false; // check select gamestart
  int stage=1; // parameter for checking stage
};
```

02 단계별 수행 과정

□ 1단계

목표 | Ncurses library 함수들을 사용해서 2차원 배열로 된 Map을 Game 화면으로 표시하기

```
2111111111111111111111111111111111111112
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
1000000000000000000000000000000000000001
2111111111111111111111111111111111111112
```

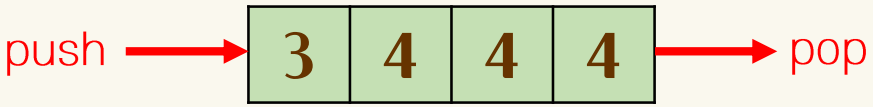


※ 03 추가한 규칙 5에 의해 앞으로는
그래픽 추가한 사진을 사용할 예정

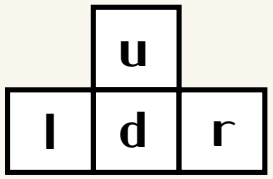
□ 2단계

목표 | 1단계의 Map 위에 Snake를 표시하고 화살표를 입력받아 Snake가 움직이게 하기

1. Snake의 구현 | FIFO(First In First Out)를 위해 queue로 구현



2. 키보드로 방향 입력받기 | getch( ) 함수

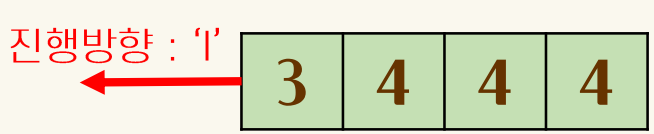


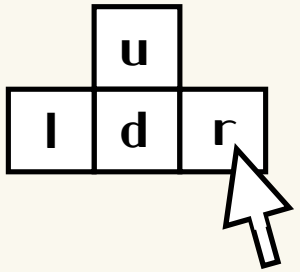
□ 2단계

목표 | 1단계의 Map 위에 Snake를 표시하고 화살표를 입력받아 Snake가 움직이게 하기

3. Snake가 진행 방향의 반대 방향으로 이동했을 경우 | WrongDirection( ) 함수

ex >





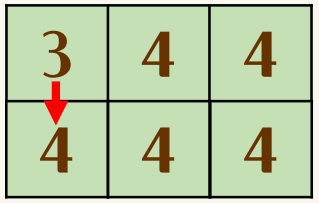
→ WrongDirection( ) 함수 실행

□ 2단계

목표 | 1단계의 Map 위에 Snake를 표시하고 화살표를 입력받아 Snake가 움직이게 하기

4. Snake가 자신의 body를 만났을 경우 | BumpedintoBody( ) 함수

ex >



→ BumpedintoBody( ) 함수 실행

□ 2단계

목표 | 1단계의 Map 위에 Snake를 표시하고 화살표를 입력받아 Snake가 움직이게 하기

5. Snake가 Wall과 만났을 경우 | BumpedintoWall( ) 함수

ex >

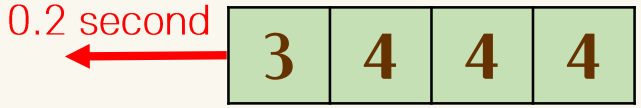
→ BumpedintoWall( ) 함수 실행

□ 2단계

목표 | 1단계의 Map 위에 Snake를 표시하고 화살표를 입력받아 Snake가 움직이게 하기

6. 0.2초마다 Snake 움직이게 하기 | usleep( ) 함수

ex >

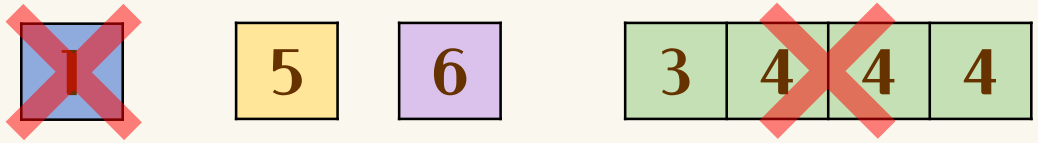


→ usleep(2000000)

□ 3단계

목표 | 2단계 프로그램에서 Map 위에 Growth item과 Poison item이 출현하도록 하기

1. Growth item과 Poison item 랜덤으로 생성하기 | RandGrowthItem( ), RandPoisonItem( ) 함수



→ wall[x][y]!=1

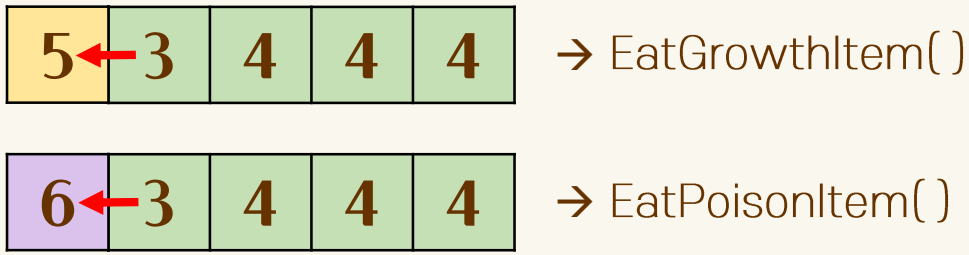
→ IsBody(x,y)==false

□ 3단계

목표 | 2단계 프로그램에서 Map 위에 Growth item과 Poison item이 출현하도록 하기

2. Snake의 head가 item과 만나는지 확인하기 | EatGrowthItem( ), EatPoisonItem( ) 함수

ex >



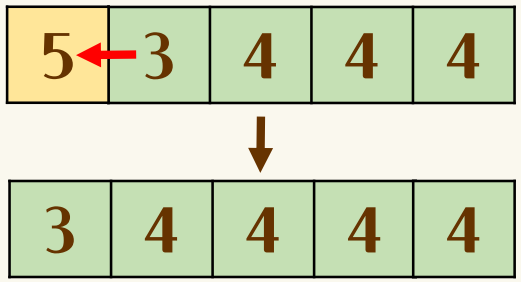
□ 3단계

목표 | 2단계 프로그램에서 Map 위에 Growth item과 Poison item이 출현하도록 하기

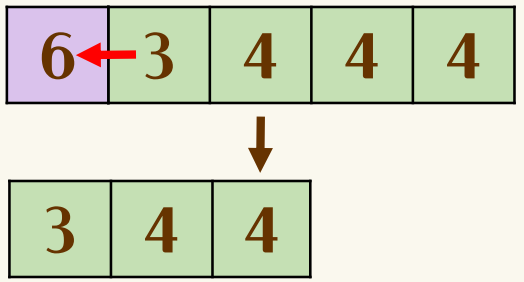
3. Growth item을 먹으면 길이 +1 / Poison item을 먹으면 길이 -1

ex >

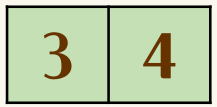
▼ EatGrowthItem( ) == true



▼ EatPoisonItem( ) == true



▼ if(len) < 3 → SizeofSnake( ) (GAME OVER!)

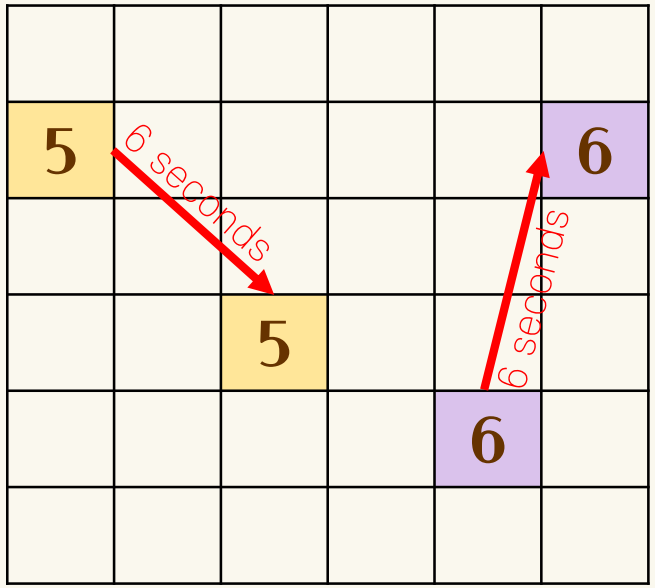


□ 3단계

목표 | 2단계 프로그램에서 Map 위에 Growth item과 Poison item이 출현하도록 하기

4. 6초마다 item의 위치 바꿔주기 | RandGrowthItem( ), RandPoisonItem( ) 함수

ex >

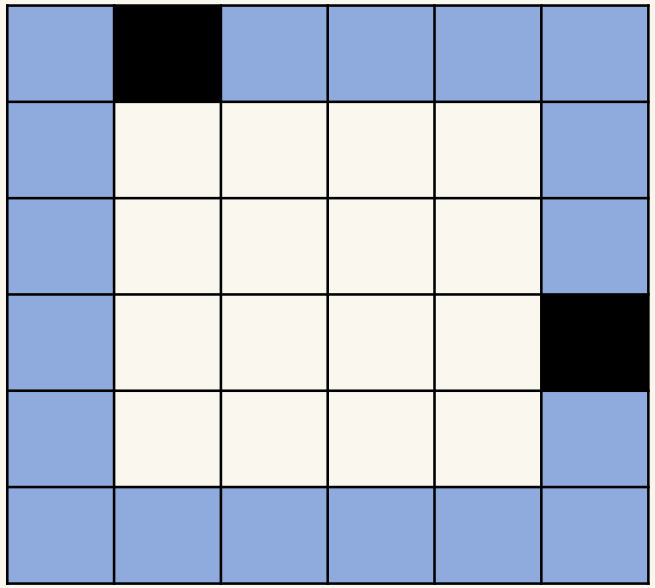


□ 4단계

목표 | 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 하기

1. Wall에 임의로 한 쌍의 Gate 만들기 | RandGate( ) 함수

ex >



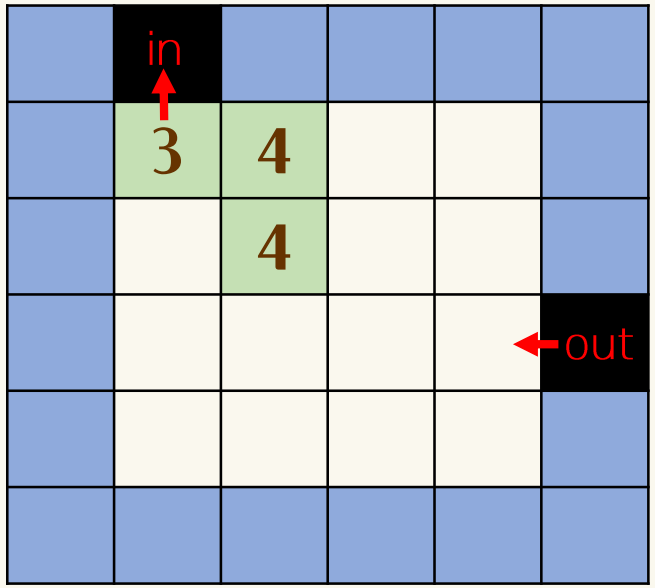
→ wall[x][y]==1

□ 4단계

목표 | 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 하기

2. Gate인지 판별하기 & in / out Gate로 구분하기 | IsGate( ) 함수

ex >



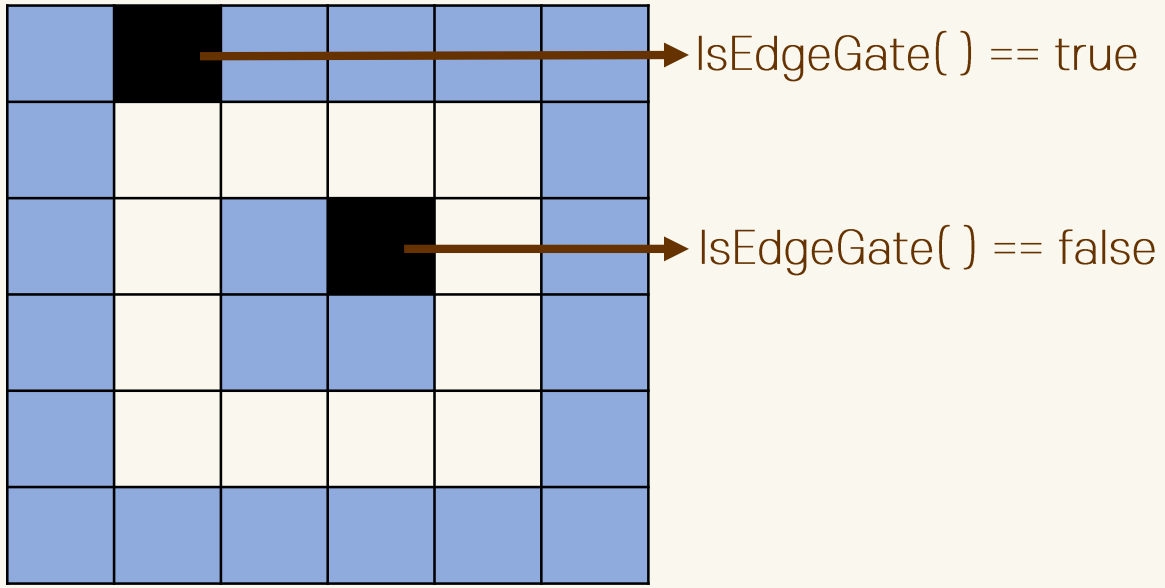
→ IsGate( ) == true

□ 4단계

목표 | 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 하기

3. Gate가 가장자리에 생기는지 판단하기 | IsEdgeGate( ) 함수

ex >



□ 4단계

목표 | 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 하기

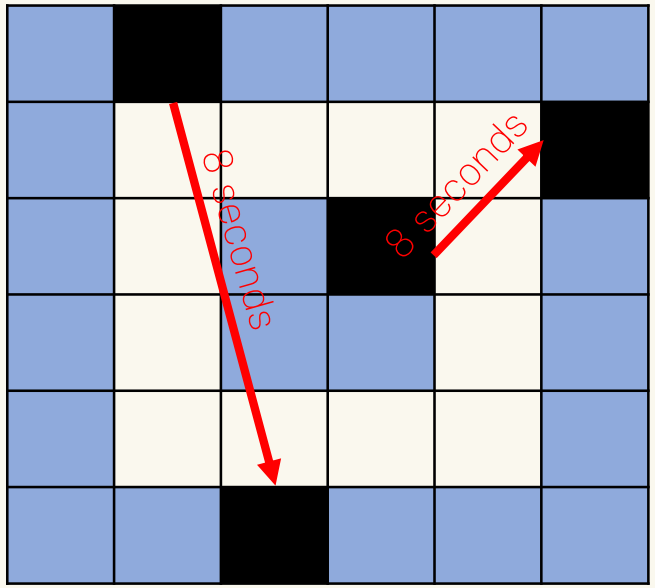
4. Gate를 지날 때 Gate의 위치에 따라 Snake가 움직이게 하기 | ControlGate( ) 함수

□ 4단계

목표 | 3단계 프로그램에서 Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 하기

5. 8초마다 Gate 위치 바꿔주기 | RandGate( ) 함수

ex >



□ 5단계

목표 | 4단계 프로그램에서 우측에 게임 점수를 표시하는 화면이 뜨도록 만들기
| 각 stage의 mission을 달성하면 다음 Map으로 진행하도록 하기

1. Scoreboard 만들기 | RandGate( ) 함수

ex >

〈Score Board〉
B : 4 / 7
+ : 2 / 3
- : 1 / 2
G : 0 / 1

□ 5단계

목표 | 4단계 프로그램에서 우측에 게임 점수를 표시하는 화면이 뜨도록 만들기
| 각 stage의 mission을 달성하면 다음 Map으로 진행하도록 하기

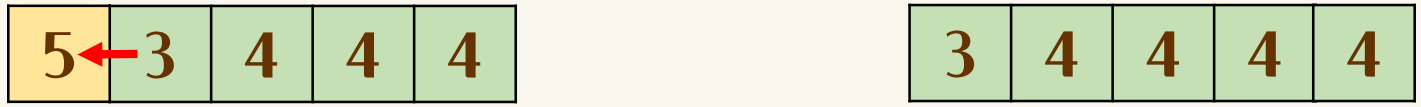
2. Scoreboard 갱신 | ScoreBoard( ) 함수

ex >

〈Score Board〉
B : 4 / 7
+ : 2 / 3
- : 1 / 2
G : 0 / 1

ScoreBoard( )

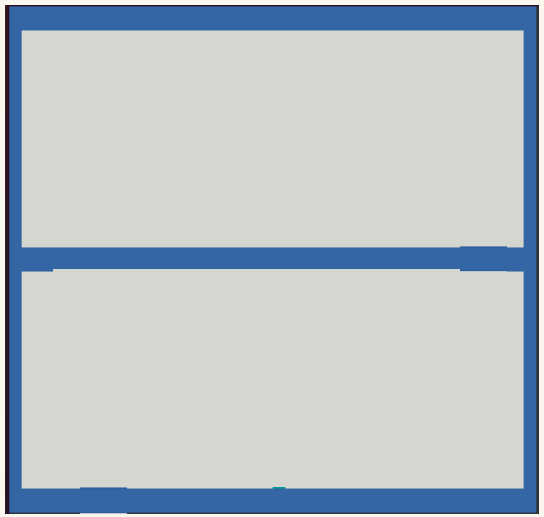
〈Score Board〉
B : 5 / 7
+ : 3 / 3
- : 1 / 2
G : 0 / 1



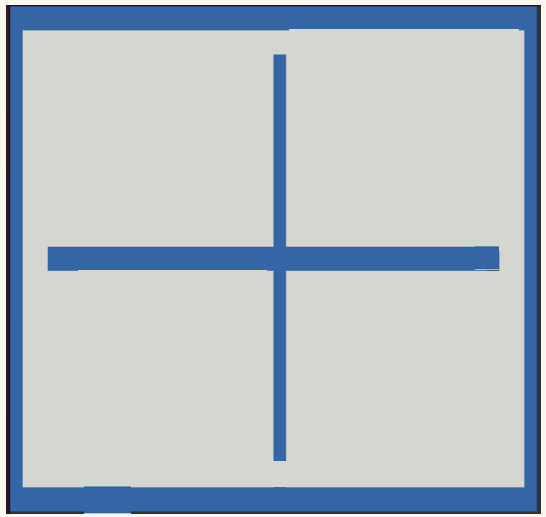
□ 5단계

목표 | 4단계 프로그램에서 우측에 게임 점수를 표시하는 화면이 뜨도록 만들기
| 각 stage의 mission을 달성하면 다음 Map으로 진행하도록 하기

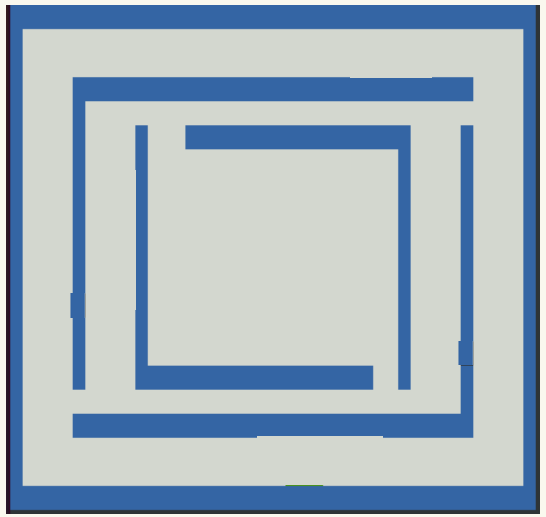
3. Stage 별로 map 만들기 | MakeMap1( ), MakeMap2( ), MakeMap3( ) 함수



▲Map2



▲Map3



▲Map4

□ 5단계

목표 | 4단계 프로그램에서 우측에 게임 점수를 표시하는 화면이 뜨도록 만들기
| 각 stage의 mission을 달성하면 다음 Map으로 진행하도록 하기

4. 해당 Stage의 미션을 성공했는지 여부 파악하기 | CheckMission( ) 함수

| Mission parameter

* stage 1	* stage 2	* stage 3	* stage 4
B : 7	B : 8	B : 9	B : 10
+ : 3	+ : 3	+ : 4	+ : 5
- : 2	- : 3	- : 4	- : 5
G : 1	G : 2	G : 3	G : 4

□ 5단계

목표 | 4단계 프로그램에서 우측에 게임 점수를 표시하는 화면이 뜨도록 만들기
| 각 stage의 mission을 달성하면 다음 Map으로 진행하도록 하기

5. 다음 stage로 넘어가게 하기 | main.cpp

If CheckMission( ) == true → next stage

□ 1. 효과음 넣기

목표 | 게임이 종료된 경우나 다음 stage로 넘어가는 경우 효과음 넣기

게임이 종료된 경우, 다음 stage로 넘어가는 경우

→ beep( ) 함수 사용

□ 2. Snake를 랜덤하게 놓기

목표 | stage 1이 시작할 때 Snake가 Wall을 제외한 Map 위에 랜덤하게 놓이게 하기

```
272    if(stage == 1)
273    {
274      srand((unsigned int)time(0));
275      Randwidth = rand() % (width-6)+3;
276      Randheight = rand() % (height-6)+3;
277      int c=rand()%4;
278
279      direction=' ';
280
281      if(c==0)
282      {
283        mvwprintw(stdscr,Randheight,Randwidth,"3");
284        mvwprintw(stdscr,Randheight,Randwidth+1,"4");
285        mvwprintw(stdscr,Randheight,Randwidth+2,"4");
286        xy.push(make_pair(Randheight,Randwidth+2));
287        xy.push(make_pair(Randheight,Randwidth+1));
288        xy.push(make_pair(Randheight,Randwidth));
289        direction='l'; // left
290      }
291       if(c==1)
292       {
293         mvwprintw(stdscr,Randheight,Randwidth,"3");
294         mvwprintw(stdscr,Randheight,Randwidth-1,"4");
295         mvwprintw(stdscr,Randheight,Randwidth-2,"4");
296         xy.push(make_pair(Randheight,Randwidth-2));
297         xy.push(make_pair(Randheight,Randwidth-1));
298         xy.push(make_pair(Randheight,Randwidth));
299         direction='r'; // right
300       }
```

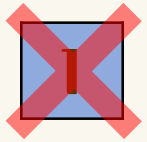
```
301       if(c==2)
302       {
303         mvwprintw(stdscr,Randheight,Randwidth,"3");
304         mvwprintw(stdscr,Randheight+1,Randwidth,"4");
305         mvwprintw(stdscr,Randheight+2,Randwidth,"4");
306         xy.push(make_pair(Randheight+2,Randwidth));
307         xy.push(make_pair(Randheight+1,Randwidth));
308         xy.push(make_pair(Randheight,Randwidth));
309         direction='u'; // up
310       }
311      if(c==3)
312      {
313        mvwprintw(stdscr,Randheight,Randwidth,"3");
314        mvwprintw(stdscr,Randheight-1,Randwidth,"4");
315        mvwprintw(stdscr,Randheight-2,Randwidth,"4");
316        xy.push(make_pair(Randheight-2,Randwidth));
317        xy.push(make_pair(Randheight-1,Randwidth));
318        xy.push(make_pair(Randheight,Randwidth));
319        direction='d'; // down
320      }
321    }
```

□ 3. 랜덤으로 생기는 Hole

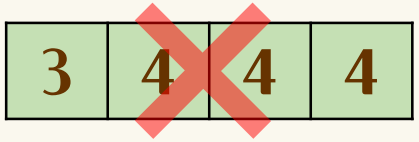
03 추가한 규칙

목표 | Hole을 랜덤하게 생성하여 Snake와 Hole이 만나면 게임을 종료하도록 하기

1. Hole 랜덤하게 생성하기 | RandHole( ) 함수



→ wall[x][y]!=1



→ IsBody(x,y)==false

□ 3. 랜덤으로 생기는 Hole

목표 | Hole을 랜덤하게 생성하여 Snake와 Hole이 만나면 게임을 종료하도록 하기

2. Snake의 Head가 Hole과 만나는지 확인하기 | IsHole( ) 함수

→ IsHole( ) == true

□ 3. 랜덤으로 생기는 Hole

03 추가한 규칙

목표 | Hole을 랜덤하게 생성하여 Snake와 Hole이 만나면 게임을 종료하도록 하기

3. Snake의 Head가 Hole과 만나면 게임 종료하기 | BumpedintoHole( ) 함수

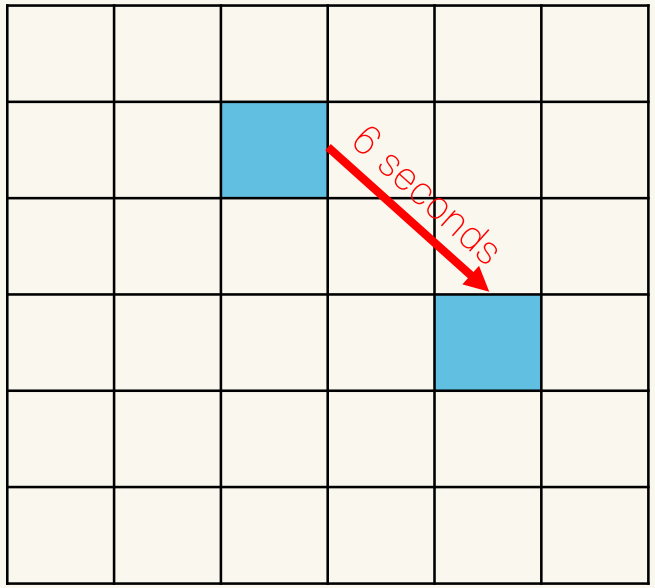
IsHole( ) == true → BumpedintoHole( ) (GAME OVER!)

□ 3. 랜덤으로 생기는 Hole

03 추가한 규칙

목표 | Hole을 랜덤하게 생성하여 Snake와 Hole이 만나면 게임을 종료하도록 하기

4. 6초마다 Hole의 위치 바꿔주기 | RandHole( ) 함수

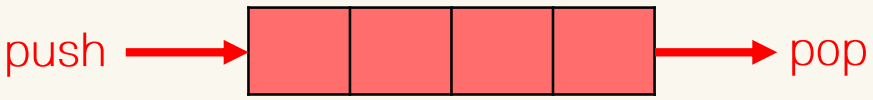


□ 4. Map을 따라 도는 Cobra

03 추가한 규칙

목표 | Snake가 Map을 따라 일정하게 도는 Cobra와 만나면 게임을 종료하도록 하기

1. Cobra 구현하기 | FIFO(First In First Out)를 위해 queue로 구현



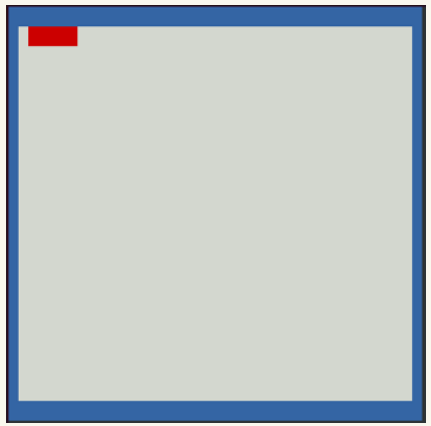
- Cobra 길이는 5로 고정

□ 4. Map을 따라 도는 Cobra

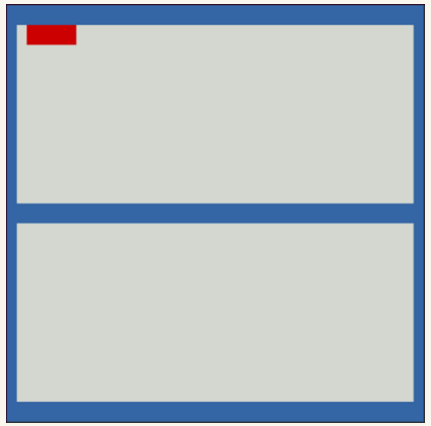
03 추가한 규칙

목표 | Snake가 Map을 따라 일정하게 도는 Cobra와 만나면 게임을 종료하도록 하기

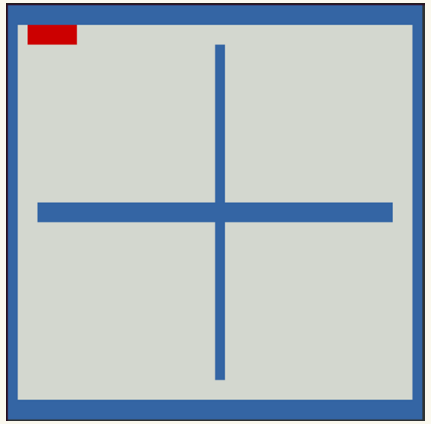
2. Cobra의 초기 위치를 잡아주기 | LocateCobra( ) 함수



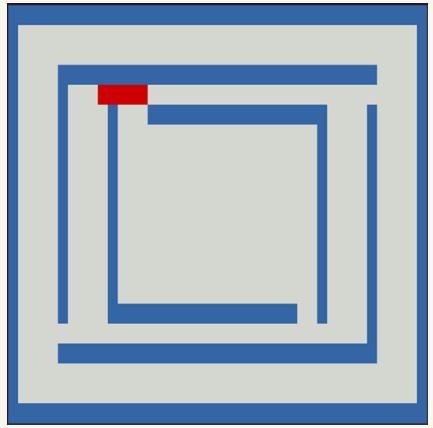
▲Map1



▲Map2



▲Map3

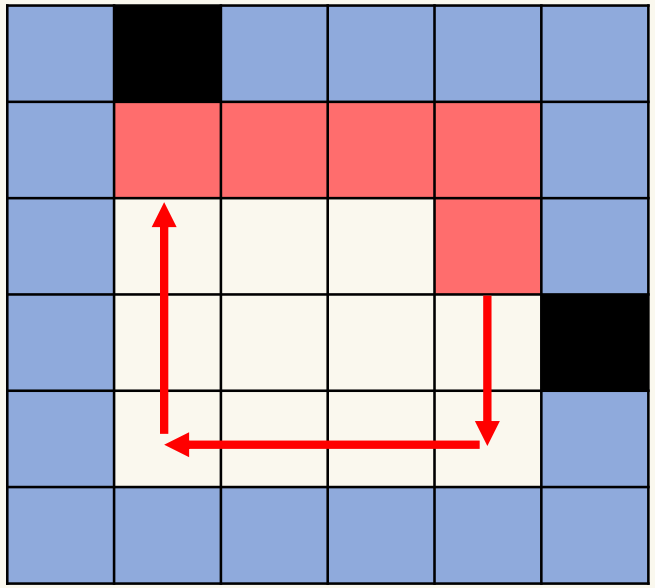


▲Map4

□ 4. Map을 따라 도는 Cobra

목표 | Snake가 Map을 따라 일정하게 도는 Cobra와 만나면 게임을 종료하도록 하기

3. Cobra가 Map을 따라 돌도록 만들기 | DangerousCobra( ) 함수



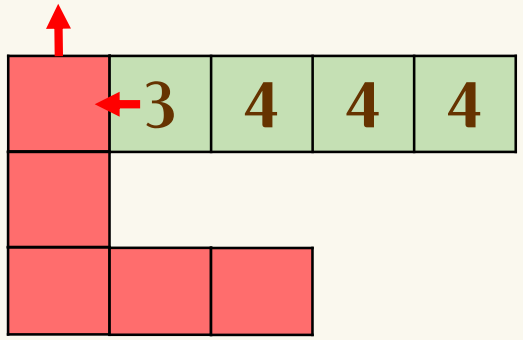
□ 4. Map을 따라 도는 Cobra

03 추가한 규칙

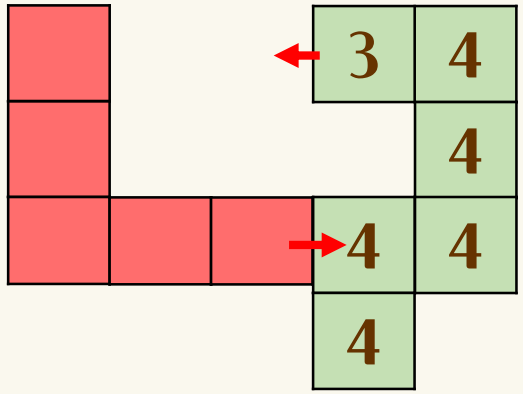
목표 | Snake가 Map을 따라 일정하게 도는 Cobra와 만나면 게임을 종료하도록 하기

4. Snake가 Cobra와 만나는지 확인하기 | IsCobra( ) 함수

ex >



→ IsCobra( ) == true



→ IsCobra( ) == true

□ 4. Map을 따라 도는 Cobra

03 추가한 규칙

목표 | Snake가 Map을 따라 일정하게 도는 Cobra와 만나면 게임을 종료하도록 하기

5. Snake가 Cobra와 만나면 게임 종료하기 | BumpedintoCobra( ) 함수

IsCobra( ) == true → BumpedintoCobra( ) (GAME OVER!)

□ 5. 그래픽 추가하기

목표 | 게임하기 편하도록 색을 달리하여 구분 가능하게 만들기

| init_pair( ), attron( ), attroff( ) 함수

□ 1. 실행하기

목표 | g++ 컴파일 하고 make 파일로 컴파일하기

1. g++ 컴파일 하기 | g++ -o snake make.cpp SnakeGame.cpp -lncurses

2. Make 파일 | make

□ 2. 개선할 점

1. Cobra가 경로 이탈하는 현상이 발생하는 점

2. Game over가 된 이후 창이 제대로 닫히지 않는 점

감사합니다 :-)