KIB_C_game_project_snake

다음에 의해 생성됨 : Doxygen 1.9.1

Chapter 1

KIB_c_game_project_snake

kib!! 최수길 왔다감 두번째 왔다감. 세번째 왔다감.

Chapter 2

파일 색인

2.1 파일 목록

다음은 모든 파일에 대한 목록입니다. (간략한 설명만을 보여줍니다) :

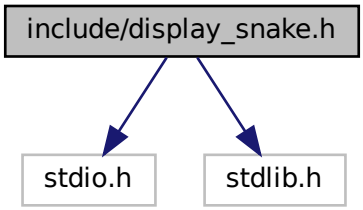
Chapter 3

파일 문서화

3.1 include/display_snake.h 파일 참조

```
#include <stdio.h>
#include <stdlib.h>
```

display_snake.h에 대한 include 의존 그래프



함수

- void display_snake ()

  display_snake(필드구성함수)를 정의했습니다 게임을 시작했을때 기존의 화면을 지우고 새로운 화면을 만들기 위한 system(clear)함수를 사용하였고 게임 영역의 상단을 나타내는 '@' 문자들을 출력합니다. 또한 이중 루프를 사용하여 게임 영역의 각 셀을 반복하면서

3.1.1 함수 문서화

3.1.1.1 display_snake()

```
void display_snake ( )
```

display_snake(필드구성함수)를 정의했습니다 게임을 시작했을때 기존의 화면을 지우고 새로운 화면을 만들기 위한 system(clear)함수를 사용하였고 게임 영역의 상단을 나타내는 '@' 문자들을 출력합니다. 또한 이중 루프를 사용하여 게임 영역의 각 셀을 반복하면서

왼쪽 벽('@' 문자)을 출력합니다. 현재 셀이 뱀의 머리인 경우 'O'를 출력합니다. 현재 셀이 먹이인 경우 'P'를 출력합니다. 그렇지 않은 경우: 뱀 꼬리가 존재하는지 확인하고, 꼬리가 있는 경우 'o'를 출력합니다. 꼬리가 없는 경우 빈 공간을 출력합니다. 오른쪽 벽('@' 문자)을 출력합니다. 현재 점수와 시간, 꼬리의 길이를 실시간으로 표시합니다.

display_snake.c 파일의 44 번째 라인에서 정의되었습니다.

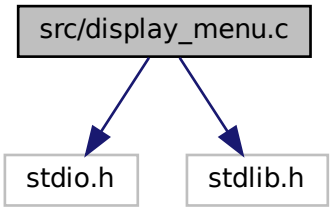
```
45 {
46     system("clear");
47     // ceiling print
48     for (int i = 0; i < height; ++i)
49     {
50         for (int j = 0; j < width; ++j)
51         {
52             for (int k = 0; k < entireTail; ++k)
53             {
54                 if (i == snakeX[k] && j == snakeY[k])
55                 {
56                     printf("o");
57                 }
58             }
59             if (i == 0 || i == height - 1)
60             {
61                 printf("@");
62             }
63             else if (j == 0 || j == width - 1)
64             {
65                 printf("@");
66             }
67             else if (i == x && j == y)
68             {
69                 printf("1");
70             }
71             else if (i == baitX && j == baitY)
72             {
73                 printf("2");
74             }
75             else
76                 printf(" ");
77         }
78         printf("\n");
79     }
80     printf("\n\n");
81     printf("Your Point : %d\n", point);
82     printf("\n\n");
83     printf("Timer : %d\n", ++count);
84     printf("\n\n");
85     printf("tail : %d\n", entireTail);
86 }
```

3.2 README.md 파일 참조

3.3 src/display_menu.c 파일 참조

```
#include <stdio.h>
#include <stdlib.h>
```

display_menu.c에 대한 include 의존 그래프



함수

- int display_menu ()

  display_menu함수는 menu변수를 정의하고 화면을 초기하고 게임메뉴들을 표시해주는 함수입니다.

3.3.1 함수 문서화

3.3.1.1 display_menu()

```
int display_menu ( )
```

display_menu함수는 menu변수를 정의하고 화면을 초기하고 게임메뉴들을 표시해주는 함수입니다.

display_menu.c 파일의 13 번째 라인에서 정의되었습니다.

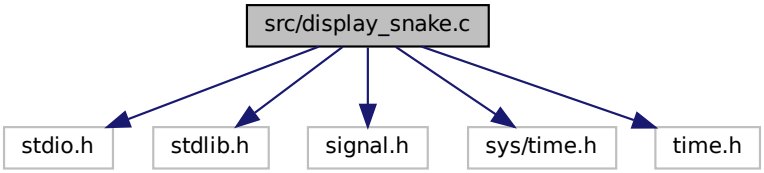
```
14 {
15     int menu;
16 
17      system("clear");
18 
19 
20     printf("\n\n\n");
21 
22     printf("  SSSS  N   N   AAA   K   k  EEEEE    GGG    AAA   M   M  EEEEE\n");
23     printf(" S      NN  N  A   A  K  K   E       G      A   A  MM MM  E    \n");
24     printf("  SSS   N N N  AAAAA  KK     EEEE    GGGG   AAAAA  M M M  EEEE \n");
25     printf("     S  N  NN  A   A  K  K   E      G    G  A   A  M   M  E    \n");
26     printf(" SSSS   N   N  A   A  K   K  EEEEE   GGGG   A   A  M   M  EEEEE\n");
27 
28 
29 
30 
31     printf("\tWelcome to Snake Game!\n");
32 
33     printf("-------------------------------------------------------\n");
34     printf("\tAre You Ready To Play The Snake Game?\n");
35     printf("-------------------------------------------------------\n");
36     printf("\t\tGAME MENU\n");
37     printf("-------------------------------------------------------\n");
38     printf("\t\t1. Game Start\n");
39     printf("\t\t2. Record\n");
40     printf("\t\t3. Quit\n");
41     printf("-------------------------------------------------------\n");
42 
43     printf("Press any key to start...\n");
44 
45     scanf("%d", &menu);
46     return menu;
47 }
```

3.4 src/display_snake.c 파일 참조

printf,signal,clear함수와 count변수와 시간을 표시하기 위한 헤더파일을 표시했습니다.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/time.h>
#include <time.h>
```

display_snake.c에 대한 include 의존 그래프



매크로

- #define width 40
- #define height 40

함수

- void display_snake ()

  display_snake(필드구성함수)를 정의했습니다 게임을 시작했을때 기존의 화면을 지우고 새로운 화면을 만들기 위한 system(clear)함수를 사용하였고 게임 영역의 상단을 나타내는 '@' 문자들을 출력합니다. 또한 이중 루프를 사용하여 게임 영역의 각 셀을 반복하면서

변수

- int count
- int x
- int y
- int baitX
- int baitY
- int entireTail
- int point
- int snakeX [100]
- int snakeY [100]

3.4.1 상세한 설명

printf,signal,clear함수와 count변수와 시간을 표시하기 위한 헤더파일을 표시했습니다.

플레이어의 뱀의 기본크기를 정의하였고 먹이의 크기 필드의크기 포인트 꼬리의전체크기변수와 꼬리의 변화배열을 가져와 표시했습니다.

3.4.2 매크로 문서화

3.4.2.1 height

```
#define height 40
```

display_snake.c 파일의 16 번째 라인에서 정의되었습니다.

3.4.2.2 width

```
#define width 40
```

display_snake.c 파일의 15 번째 라인에서 정의되었습니다.

3.4.3 함수 문서화

3.4.3.1 display_snake()

```
void display_snake ( )
```

display_snake(필드구성함수)를 정의했습니다 게임을 시작했을때 기존의 화면을 지우고 새로운 화면을 만들기 위한 system(clear)함수를 사용하였고 게임 영역의 상단을 나타내는 '@' 문자들을 출력합니다. 또한 이중 루프를 사용하여 게임 영역의 각 셀을 반복하면서

왼쪽 벽('@' 문자)을 출력합니다. 현재 셀이 뱀의 머리인 경우 'O'를 출력합니다. 현재 셀이 먹이인 경우 'P'를 출력합니다. 그렇지 않은 경우: 뱀 꼬리가 존재하는지 확인하고, 꼬리가 있는 경우 'o'를 출력합니다. 꼬리가 없는 경우 빈 공간을 출력합니다. 오른쪽 벽('@' 문자)을 출력합니다. 현재 점수와 시간, 꼬리의 길이를 실시간으로 표시합니다.

display_snake.c 파일의 44 번째 라인에서 정의되었습니다.

```
45 {
46     system("clear");
47     // ceiling print
48     for (int i = 0; i < height; ++i)
49     {
50         for (int j = 0; j < width; ++j)
51         {
52             for (int k = 0; k < entireTail; ++k)
53             {
54                 if (i == snakeX[k] && j == snakeY[k])
55                 {
56                     printf("o");
57                 }
58             }
59             if (i == 0 || i == height - 1)
60             {
61                 printf("@");
62             }
63             else if (j == 0 || j == width - 1)
64             {
65                 printf("@");
66             }
```

```
67              else if (i == x && j == y)
68              {
69                  printf("1");
70              }
71              else if (i == baitX && j == baitY)
72              {
73                  printf("2");
74              }
75              else
76                  printf(" ");
77          }
78          printf("\n");
79      }
80      printf("\n\n");
81      printf("Your Point : %d\n", point);
82      printf("\n\n");
83      printf("Timer : %d\n", ++count);
84      printf("\n\n");
85      printf("tail : %d\n", entireTail);
86  }
```

3.4.4 변수 문서화

3.4.4.1 baitX

```
int baitX  [extern]
```

update.c 파일의 27 번째 라인에서 정의되었습니다.

3.4.4.2 baitY

```
int baitY  [extern]
```

update.c 파일의 28 번째 라인에서 정의되었습니다.

3.4.4.3 count

```
int count  [extern]
```

update.c 파일의 20 번째 라인에서 정의되었습니다.

3.4.4.4 entireTail

```
int entireTail  [extern]
```

main.c 파일의 24 번째 라인에서 정의되었습니다.

3.4.4.5 point

```
int point  [extern]
```

main.c 파일의 21 번째 라인에서 정의되었습니다.

3.4.4.6 snakeX

```
int snakeX[100]  [extern]
```

update.c 파일의 29 번째 라인에서 정의되었습니다.

3.4.4.7 snakeY

```
int snakeY[100]  [extern]
```

update.c 파일의 30 번째 라인에서 정의되었습니다.

3.4.4.8 x

```
int x  [extern]
```

main.c 파일의 22 번째 라인에서 정의되었습니다.

3.4.4.9 y

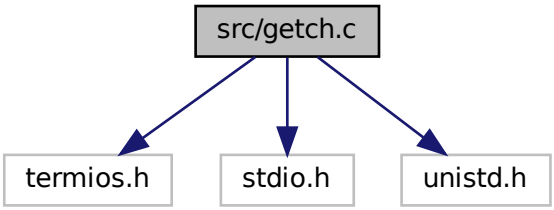
```
int y  [extern]
```

main.c 파일의 23 번째 라인에서 정의되었습니다.

3.5 src/getch.c 파일 참조

```
#include <termios.h>
#include <stdio.h>
#include <unistd.h>
```

getch.c에 대한 include 의존 그래프



매크로

- #define CCHAR 0
- #define CTIME 1
- #define FLAG TCSAFLUSH

함수

- int getch (void)

3.5.1 매크로 문서화

3.5.1.1 CCHAR

```
#define CCHAR 0
```

getch.c 파일의 6 번째 라인에서 정의되었습니다.

3.5.1.2 CTIME

```
#define CTIME 1
```

getch.c 파일의 10 번째 라인에서 정의되었습니다.

3.5.1.3 FLAG

```
#define FLAG TCSAFLUSH
```

3.5.2 함수 문서화

3.5.2.1 getch()

```
int getch (
              void  )
```

getch.c 파일의 13 번째 라인에서 정의되었습니다.

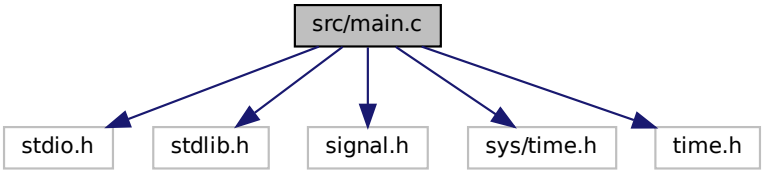
```
14 {
15     char ch;
16     int error;
17     static struct termios Otty, Ntty;
18 
19     //  fflush(stdout);
20     tcgetattr(0, &Otty);
21     Ntty = Otty;
22     Ntty.c_iflag = 0;
23     Ntty.c_oflag = 0;
24     Ntty.c_lflag &= ~ICANON;
25 #if 1
26     Ntty.c_lflag &= ~ECHO;
27 #else
28     Ntty.c_lflag |= ECHO;
29 #endif
30     Ntty.c_cc[VMIN] = CCHAR;
31     Ntty.c_cc[VTIME] = CTIME;
32 
33 #if 1
34 #define FLAG TCSAFLUSH
35 #else
36 #define FLAG TCSANOW
37 #endif
38 
39     if (0 == (error = tcsetattr(0, FLAG, &Ntty)))
40     {
41         error = read(0, &ch, 1);
42         error += tcsetattr(0, FLAG, &Otty);
43     }
44 
45     return (error == 1 ? (int)ch : -1);
46 }
```

3.6 src/main.c 파일 참조

printf함수, signal함수, 타이머, static struct를 활용하기 위한 헤더파일을 포함시켰습니다. game,point, 필드의 크기, 처음 시작했을때 꼬리의길이, 꼬리의 변화를 주는 변수, 게임 시작과 끝을 정의하고 미끼의 위치,snakedirection(sdir), update변수, disdplay_menu변수, display_snake변수를 가져와서 사용했습니다.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/time.h>
```

```
#include <time.h>
```
main.c에 대한 include 의존 그래프



매크로

- #define GAME_START 0
- #define GAME_END 1
- #define width 40
- #define height 40

함수

- void space ()
- int update (int signum)
- int display_menu ()

  display_menu함수는 menu변수를 정의하고 화면을 초기하고 게임메뉴들을 표시해주는 함수입니다.
- void display_snake ()

  display_snake(필드구성함수)를 정의했습니다 게임을 시작했을때 기존의 화면을 지우고 새로운 화면을 만들기 위한 system(clear)함수를 사용하였고 게임 영역의 상단을 나타내는 '@' 문자들을 출력합니다. 또한 이중 루프를 사용하여 게임 영역의 각 셀을 반복하면서
- int game_start ()

  game_start 함수를 정의하였습니다. itimerval 구조체는 POSIX 시스템에서 타이머 값을 설정하고 제어하는 데 사용됩니다. 이 구조체는 시간 간격을 나타내는 두 개의 필드(it.value it.interval)을 포함하며 이 구조체를 정적변수(static struct)화 시켜서 정의했습니다. 또한 signal함수를 사용하여 가상타이머가 끝나면(SIGVTALRM) update함수를 불러오도록 설정하였습니다. 또한 setitmer 함수를 사용하여 타이머를 설정하였고 타이머의 종류(ITIMER_↩ VIRTUAL(가상 타이머)),timer 구조체의 주소(&timer), 이전 타이머의 설정(보통 NULL로 표현한다.) 으로 구성했습니다. 그래서 게임이 끝났을경우 업데이트가 더이상 진행되지 않도록 신호를 보내고 게임을 다시 시작하기 위한 함수(reset)를 불러와서 게임이 끝났다는것(return = 1 <- Game END)을 알릴수 있도록 하였습니다.
- int main (void)
- void reset ()

변수

- int game
- int point = 0
- int x = width / 2
- int y = height / 2
- int entireTail
- sdir
- int snakeX [100]
- int snakeY [100]

3.6.1 상세한 설명

printf함수, signal함수, 타이머, static struct를 활용하기 위한 헤더파일을 포함시켰습니다. game,point, 필드의 크기, 처음 시작했을때 꼬리의길이, 꼬리의 변화를 주는 변수, 게임 시작과 끝을 정의하고 미끼의 위치,snakedirection(sdir), update변수, disdplay_menu변수, display_snake변수를 가져와서 사용했습니다.

3.6.2 매크로 문서화

3.6.2.1 GAME_END

```
#define GAME_END 1
```

main.c 파일의 16 번째 라인에서 정의되었습니다.

3.6.2.2 GAME_START

```
#define GAME_START 0
```

main.c 파일의 15 번째 라인에서 정의되었습니다.

3.6.2.3 height

```
#define height 40
```

main.c 파일의 18 번째 라인에서 정의되었습니다.

3.6.2.4 width

```
#define width 40
```

main.c 파일의 17 번째 라인에서 정의되었습니다.

3.6.3 함수 문서화

3.6.3.1 display_menu()

```
int display_menu ( )
```

display_menu함수는 menu변수를 정의하고 화면을 초기하고 게임메뉴들을 표시해주는 함수입니다.

display_menu.c 파일의 13 번째 라인에서 정의되었습니다.

```
14 {
15     int menu;
16 
17      system("clear");
18 
19 
20     printf("\n\n\n");
21 
22     printf("  SSSS  N   N   AAA   K   k  EEEEE    GGG    AAA   M   M  EEEEE\n");
23     printf(" S      NN  N  A   A  K  K   E       G      A   A  MM MM  E    \n");
24     printf("  SSS   N N N  AAAAA  KK     EEEE    GGGG   AAAAA  M M M  EEEE \n");
25     printf("     S  N  NN  A   A  K  K   E       G   G  A   A  M   M  E    \n");
26     printf(" SSSS   N   N  A   A  K   K  EEEEE   GGGG   A   A  M   M  EEEEE\n");
27 
28 
29 
30 
31     printf("\tWelcome to Snake Game!\n");
32 
33     printf("-------------------------------------------------------\n");
34     printf("\tAre You Ready To Play The Snake Game?\n");
35     printf("-------------------------------------------------------\n");
36     printf("\t\tGAME MENU\n");
37     printf("-------------------------------------------------------\n");
38     printf("\t\t1. Game Start\n");
39     printf("\t\t2. Record\n");
40     printf("\t\t3. Quit\n");
41     printf("-------------------------------------------------------\n");
42 
43     printf("Press any key to start...\n");
44 
45     scanf("%d", &menu);
46     return menu;
47 }
```

3.6.3.2 display_snake()

```
void display_snake ( )
```

display_snake(필드구성함수)를 정의했습니다 게임을 시작했을때 기존의 화면을 지우고 새로운 화면을 만들기 위한 system(clear)함수를 사용하였고 게임 영역의 상단을 나타내는 '@' 문자들을 출력합니다. 또한 이중 루프를 사용하여 게임 영역의 각 셀을 반복하면서

왼쪽 벽('@' 문자)을 출력합니다. 현재 셀이 뱀의 머리인 경우 'O'를 출력합니다. 현재 셀이 먹이인 경우 'P'를 출력합니다. 그렇지 않은 경우: 뱀 꼬리가 존재하는지 확인하고, 꼬리가 있는 경우 'o'를 출력합니다. 꼬리가 없는 경우 빈 공간을 출력합니다. 오른쪽 벽('@' 문자)을 출력합니다. 현재 점수와 시간, 꼬리의 길이를 실시간으로 표시합니다.

display_snake.c 파일의 44 번째 라인에서 정의되었습니다.

```
45 {
46     system("clear");
47     // ceiling print
48     for (int i = 0; i < height; ++i)
49     {
50         for (int j = 0; j < width; ++j)
51         {
52             for (int k = 0; k < entireTail; ++k)
53             {
54                 if (i == snakeX[k] && j == snakeY[k])
55                 {
56                     printf("o");
57                 }
58             }
59             if (i == 0 || i == height - 1)
```

```
60              {
61                  printf("@");
62              }
63              else if (j == 0 || j == width - 1)
64              {
65                  printf("@");
66              }
67              else if (i == x && j == y)
68              {
69                  printf("1");
70              }
71              else if (i == baitX && j == baitY)
72              {
73                  printf("2");
74              }
75              else
76                  printf(" ");
77          }
78          printf("\n");
79      }
80      printf("\n\n");
81      printf("Your Point : %d\n", point);
82      printf("\n\n");
83      printf("Timer : %d\n", ++count);
84      printf("\n\n");
85      printf("tail : %d\n", entireTail);
86  }
```

3.6.3.3 game_start()

```
int game_start ( )
```

game_start 함수를 정의하였습니다. itimerval 구조체는 POSIX 시스템에서 타이머 값을 설정하고 제어하는 데 사용됩니다. 이 구조체는 시간 간격을 나타내는 두 개의 필드(it.value it.interval)을 포함하며 이 구조체를 정적변수(static struct)화 시켜서 정의했습니다. 또한 signal함수를 사용하여 가상타이머가 끝나면(SIGVTALRM) update함수를 불러오도록 설정하였습니다. 또한 setitmer 함수를 사용하여 타이머를 설정하였고 타이머의 종류(ITIMER↩_VIRTUAL(가상 타이머)),timer 구조체의 주소(&timer), 이전 타이머의 설정(보통 NULL로 표현한다.) 으로 구성했습니다. 그래서 게임이 끝났을경우 업데이트가 더이상 진행되지 않도록 신호를 보내고 게임을 다시 시작하기 위한 함수(reset)를 불러와서 게임이 끝났다는것(return = 1 <- Game END)을 알릴수 있도록 하였습니다.

main.c 파일의 48 번째 라인에서 정의되었습니다.

```
49  {
50      // signal 설정
51      static struct itimerval timer;
52      signal(SIGVTALRM, update);
53      timer.it_value.tv_sec = 0;
54      timer.it_value.tv_usec = 16667;
55      timer.it_interval.tv_sec = 0;
56      timer.it_interval.tv_usec = 16667;
57      setitimer(ITIMER_VIRTUAL, &timer, NULL);
58
59      display_snake();
60      int update();
61
62      while (1){
63          if (game == GAME_END){
64              signal(SIGVTALRM, SIG_IGN);
65              //save_result(point);
66              reset();
67              return 1;
68          }
69      }
70      game = GAME_END;
71  }
```

3.6.3.4 main()

```
int main (
             void  )
```

메인함수의 구성은 메뉴의 변수를 정의하고 menu함수를 불러와서 표시하도록 설정 하였고 메뉴변수의 변화에 따라 switch case문을 사용하여 게임의 시작하거나 기록 을 표시하거나 게임을 빠져나갈수 있도록 설정했습니다.

main.c 파일의 77 번째 라인에서 정의되었습니다.

```
78 {
79     int menu = 1;
80     menu = display_menu();
81 
82     switch (menu)
83     {
84     case 1:
85         game = GAME_START;
86         display_snake();
87         reset();
88         menu = game_start();
89         break;
90     case 2:
91         printf("Record\n");
92         break;
93     case 3:
94         printf("Quit\n");
95         break;
96     return 0;
97     }
98 }
```

3.6.3.5 reset()

```
void reset ( )
```

reset함수를 정의하여 플레이어가 조종할수 있는 뱀의 시작위치를 표시하였습니다.

main.c 파일의 102 번째 라인에서 정의되었습니다.

```
103 {
104     point = 0;
105     x = width / 2;
106     y = height / 2;
107     snakeX[0] = x;
108     snakeY[0] = y;
109     space();
110 }
```

3.6.3.6 space()

```
void space ( )
```

update.c 파일의 33 번째 라인에서 정의되었습니다.

```
34 {
35     srand(time(NULL));
36 
37     baitX = rand()%height-1;
38     baitY = rand()%width-1;
39 
40     if(baitX == 0 || baitX == height)
41     {baitX = rand()%height-1;}
42     else if(baitY == 0 || baitY == width)
43     {baitY = rand()%width-1;}
44 
45     if (x == baitX && y == baitY)
46     {
47         baitX = rand()%height-1;
48         baitY = rand()%width-1;
49     }
50 
51 }
```

3.6.3.7 update()

```
int update (
            int signum )
```

뱀이 먹이를 섭취하였을경우에 생기는 꼬리의 변화를 임의의 변수 (temp)로 정의하고 배열을 활용하여 꼬리가 길어지는 상황을 조건문을 활 용하여 정의했습니다.

update.c 파일의 111 번째 라인에서 정의되었습니다.

```
112 {
113     int ch;
114     ch = getch();
115         switch (ch)
116         {
117         case 'i':
118             move_snake(UP);
119             break;
120         case 'j':
121             move_snake(LEFT);
122             break;
123         case 'k':
124             move_snake(DOWN);
125             break;
126         case 'l':
127             move_snake(RIGHT);
128             break;
129         case 's':
130             game = GAME_END;
131             break;
132         }
133 
134         ++ count;
135         display_snake();
136         collison();
137     /*
138          gameover if snake touch it's own body
139     for (int i = 1; i < entireTail; ++i)
140     {
141         if(snakeX[0]==snakeX[i] && snakeY[0]==snakeY[i])
142          game = GAME_END;
143     }
144     */
145 
146 
147     // gameover if snake touch wall
148     if (x == 0){
149         game = GAME_END;
150 
151     }
152     if (y == 0){
153         game = GAME_END;
154 
155     }
156     if (x == width-1){
157         game = GAME_END;
158 
159     }
160     if (y == height-1){
161         game = GAME_END;
162 
163     }
164 
170     // snake tail logic
171     int tempX = snakeX[0];
172     int tempY = snakeY[0];
173     int tempX2, tempY2;
174     snakeX[0] = x;
175     snakeY[0] = y;
176     for (int i = 1; i < entireTail; i++) {
177         tempX2 = snakeX[i];
178         tempY2 = snakeY[i];
179         snakeX[i] = tempX;
180         snakeY[i] = tempY;
181         tempX = tempX2;
182         tempY = tempY2;
183     }
184     return 0;
185 }
```

3.6.4 변수 문서화

3.6.4.1 entireTail

```
int entireTail
```

main.c 파일의 24 번째 라인에서 정의되었습니다.

3.6.4.2 game

```
int game
```

main.c 파일의 20 번째 라인에서 정의되었습니다.

3.6.4.3 point

```
int point = 0
```

main.c 파일의 21 번째 라인에서 정의되었습니다.

3.6.4.4 sdir

```
sdir  [extern]
```

3.6.4.5 snakeX

```
int snakeX[100]  [extern]
```

update.c 파일의 29 번째 라인에서 정의되었습니다.

3.6.4.6 snakeY

```
int snakeY[100]  [extern]
```

update.c 파일의 30 번째 라인에서 정의되었습니다.

3.6.4.7 x

```
int x = width / 2
```

main.c 파일의 22 번째 라인에서 정의되었습니다.

3.6.4.8 y

```
int y = height / 2
```

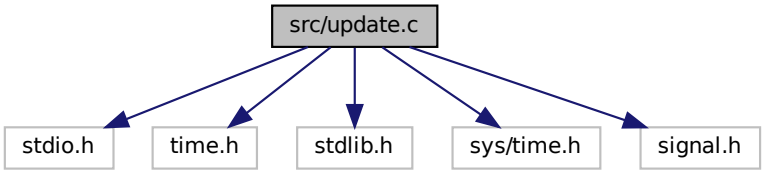
main.c 파일의 23 번째 라인에서 정의되었습니다.

3.7 src/update.c 파일 참조

rand와 srand,time함수,출력함수,signal함수를 사용하기 위한 헤더파일들을 출력했습니다. GAME START, GAME END을 정의하고 미끼의 생성,처음 생성된 미끼의 길이와 높이를 정의했습니다. 또한 게임 플레이 시간을 계산하는 count변수와 미끼의 위치 미끼를 섭취했을때의 꼬리의 변화를 정의하고(bait,snakeX,Y) 필드의 크기와 게임 포인트 꼬리의 크기변수 를 가져와서(extern) 활용하였습니다.

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <sys/time.h>
#include <signal.h>
```

update.c에 대한 include 의존 그래프



매크로

- #define GAME_START 0
- #define GAME_END 1
- #define width 40
- #define height 40

열거형 타입

- enum sdir {
  STOP = 0 , LEFT , RIGHT , UP ,
  DOWN }

함수

- void space ()
- int move_snake (int sdir)

  move_snake함수를 정의하였습니다. 위에 정한 구조체 sdir을 변수로 정하고 새로 정의한 newx,newy변수를 활용하여 switch case문으로 플레이어가 조종하는 뱀의 움직임을 정의하였습니다.
- void collison ()

  collision함수는 먹이를 플레이어가 조종하는 뱀이 먹었을때 포인트를 얻고 꼬리의 길이를 업데이트하고 새로운 먹이를 랜덤한 위치에 스폰할수 있도록 정의했습니다.
- int update (int signum)

변수

- int count
- int x
- int y
- int game
- int point
- entireTail
- int baitX
- int baitY
- int snakeX [100]
- int snakeY [100]

3.7.1 상세한 설명

rand와 srand,time함수,출력함수,signal함수를 사용하기 위한 헤더파일들을 출력했습니다. GAME START, GAME END을 정의하고 미끼의 생성,처음 생성된 미끼의 길이와 높이를 정의했습니다. 또한 게임 플레이 시간을 계산하는 count변수와 미끼의 위치 미끼를 섭취했을때의 꼬리의 변화를 정의하고(bait,snakeX,Y) 필드의 크기와 게임 포인트 꼬리의 크기변수 를 가져와서(extern) 활용하였습니다.

3.7.2 매크로 문서화

3.7.2.1 GAME_END

```
#define GAME_END 1
```

update.c 파일의 16 번째 라인에서 정의되었습니다.

3.7.2.2 GAME_START

```
#define GAME_START 0
```

update.c 파일의 15 번째 라인에서 정의되었습니다.

3.7.2.3 height

```
#define height 40
```

update.c 파일의 18 번째 라인에서 정의되었습니다.

3.7.2.4 width

```
#define width 40
```

update.c 파일의 17 번째 라인에서 정의되었습니다.

3.7.3 열거형 타입 문서화

3.7.3.1 sdir

```
enum sdir
```

열거형 멤버

STOP	
LEFT	
RIGHT	
UP	
DOWN	

update.c 파일의 56 번째 라인에서 정의되었습니다.

```
57 {
58     STOP = 0,
59     LEFT,
60     RIGHT,
61     UP,
62     DOWN
63 } sdir;
```

3.7.4 함수 문서화

3.7.4.1 collison()

```
void collison ( )
```

collision함수는 먹이를 플레이어가 조종하는 뱀이 먹었을때 포인트를 얻고 꼬리의 길이를 업데이트하고 새로운 먹이를 랜덤한 위치에 스폰할수 있도록 정의했습니다.

update.c 파일의 102 번째 라인에서 정의되었습니다.

```
103 {
104     if (x == baitX && y == baitY){
105         ++point;
106         ++entireTail;
107         space();
108     }
109 }
```

3.7.4.2 move_snake()

```
int move_snake (
              int sdir )
```

move_snake함수를 정의하였습니다. 위에 정한 구조체 sdir을 변수로 정하고 새로 정의한 newx,newy변수를 활용하여 switch case문으로 플레이어가 조종하는 뱀의 움직임을 정의하였습니다.

update.c 파일의 69 번째 라인에서 정의되었습니다.

```
70 {
71     int newx, newy;
72 
73     newx = x;
74     newy = y;
75 
76     switch (sdir){
77         case LEFT:
78             newy--;
79             break;
80         case RIGHT:
81             newy++;
82             break;
83         case DOWN:
84             newx++;
85             break;
86         case UP:
87             newx--;
88             break;
89     }
90 
91     x = newx;
92     y = newy;
93 
94     return 0;
95 }
```

3.7.4.3 space()

```
void space ( )
```

update.c 파일의 33 번째 라인에서 정의되었습니다.

```
34 {
35     srand(time(NULL));
36 
37     baitX = rand()%height-1;
38     baitY = rand()%width-1;
39 
40     if(baitX == 0 || baitX == height)
41     {baitX = rand()%height-1;}
42     else if(baitY == 0 || baitY == width)
43     {baitY = rand()%width-1;}
44 
45     if (x == baitX && y == baitY)
46     {
47         baitX = rand()%height-1;
48         baitY = rand()%width-1;
49     }
50 
51 }
```

3.7.4.4 update()

```
int update (
            int signum )
```

뱀이 먹이를 섭취하였을경우에 생기는 꼬리의 변화를 임의의 변수 (temp)로 정의하고 배열을 활용하여 꼬리가 길어지는 상황을 조건문을 활 용하여 정의했습니다.

update.c 파일의 111 번째 라인에서 정의되었습니다.

```
112 {
113     int ch;
114     ch = getch();
115         switch (ch)
116         {
117         case 'i':
118             move_snake(UP);
119             break;
120         case 'j':
121             move_snake(LEFT);
122             break;
123         case 'k':
124             move_snake(DOWN);
125             break;
126         case 'l':
127             move_snake(RIGHT);
128             break;
129         case 's':
130             game = GAME_END;
131             break;
132         }
133 
134         ++ count;
135         display_snake();
136         collison();
137     /*
138          gameover if snake touch it's own body
139     for (int i = 1; i < entireTail; ++i)
140     {
141         if(snakeX[0]==snakeX[i] && snakeY[0]==snakeY[i])
142          game = GAME_END;
143     }
144     */
145 
146 
147     // gameover if snake touch wall
148     if (x == 0){
149         game = GAME_END;
150 
151     }
152     if (y == 0){
153         game = GAME_END;
154 
155     }
156     if (x == width-1){
157         game = GAME_END;
158 
159     }
160     if (y == height-1){
161         game = GAME_END;
162 
163     }
164 
170     // snake tail logic
171     int tempX = snakeX[0];
172     int tempY = snakeY[0];
173     int tempX2, tempY2;
174     snakeX[0] = x;
175     snakeY[0] = y;
176     for (int i = 1; i < entireTail; i++) {
177         tempX2 = snakeX[i];
178         tempY2 = snakeY[i];
179         snakeX[i] = tempX;
180         snakeY[i] = tempY;
181         tempX = tempX2;
182         tempY = tempY2;
183     }
184     return 0;
185 }
```

3.7.5 변수 문서화

3.7.5.1 baitX

```
int baitX
```

update.c 파일의 27 번째 라인에서 정의되었습니다.

3.7.5.2 baitY

```
int baitY
```

update.c 파일의 28 번째 라인에서 정의되었습니다.

3.7.5.3 count

```
int count
```

update.c 파일의 20 번째 라인에서 정의되었습니다.

3.7.5.4 entireTail

```
entireTail  [extern]
```

main.c 파일의 24 번째 라인에서 정의되었습니다.

3.7.5.5 game

```
int game  [extern]
```

main.c 파일의 20 번째 라인에서 정의되었습니다.

3.7.5.6 point

```
int point  [extern]
```

main.c 파일의 21 번째 라인에서 정의되었습니다.

3.7.5.7 snakeX

```
int snakeX[100]
```

update.c 파일의 29 번째 라인에서 정의되었습니다.

3.7.5.8 snakeY

```
int snakeY[100]
```

update.c 파일의 30 번째 라인에서 정의되었습니다.

3.7.5.9 x

```
int x  [extern]
```

main.c 파일의 22 번째 라인에서 정의되었습니다.

3.7.5.10 y

```
int y  [extern]
```

main.c 파일의 23 번째 라인에서 정의되었습니다.

Chapter 4

예제 문서화

4.1 typedef함수를

사용하여 STOP LEFT RIGHT UP DOWN이란 단어의 데이터타입을 정의하고 sdir(snake direction)구조체 집합 구성요소로 정의하였습니다.

함수를

Index