Tetris

0.0.1

다음에 의해 생성됨 : Doxygen 1.9.1

Chapter 1

HM2J_c

C + MySQL

1.1 Table of Contents (목차)

1.2 Requirements

- Git
  To clone this repo `git clone https://github.com/skoler/HM2J_c.git`

```
sudo snap install git
```

- Doxygen >= 1.9.1
  For documentation

```
sudo apt install doxygen doxygen-gui graphviz
brew install --cask doxygen
```

- CMake >= 3.10
  To build this project

```
sudo apt install cmake
brew install cmake
```

- MySQL >= 8.3.0 To manage data by Database
  ```
  sudo apt install mysql-server
  sudo mysql
  ```

1.3 Usage

1.3.1 Git

1.3.1.1 Commands (명령어)

- 깃허브 원격저장소(remote repository)를 현재컴퓨터(local)에 복사(clone) 하기

```
git clone <url>
```

- 저장소 줄기(branch)들 확인하기
  - 현재 접속중인 줄기(branch)들 확인
  - 모든 줄기(branch)들 확인

```
git branch
```

- 줄기(branch) 만들기

```
git branch <branch>
```

- 줄기(branch) 삭제하기

```
git branch -d <branch>
```

- 줄기(branch) 접속(checkout)하기

```
git checkout <branch>
```

- 원격저장소 `origin`의 파일들을 현재컴퓨터에 가져와,덮어쓰기(pull) - 주의 기존파일과 충돌(conflict)할 수 있음

```
git pull origin <branch>
```

- git이 관리할 파일 추가(add)하기

```
# 하나씩 추가하기
git add <filename>
# 현재폴더(.)에 있는 파일 모두 추가하기
# 이때 .gitignore 에 있는 파일들은 제외된다
git add .
```

- git이 관리중이던(`--cached` 옵션) 파일을, 더 이상 관리하지 않도록 삭제(remove: `rm`)하기

```
git rm --cached <filename>
```

- git이 관리중인 파일 상태(status) 확인하기

```
git status
```

- git이 관리중인 파일들의 상태를, 설명(`-m` 옵션)과 함께 세이브, 저장(commit)하기

```
git commit -m "message"
```

- 줄기(branch)안에 모든 세이브 파일들(commit)을, Github 원격저장소(remote repository, name: origin)에 업로드(push)하기

```
git push origin <branch>
```

- feature/gotcha 줄기(branch)에, main 줄기를 합치기(merge). 방향 주의

```
# feature/gotcha 줄기 접속
git checkout feature/gotcha
# 현재줄기(feature/gotcha)에 main 줄기를 합치기
git merge feature/gotcha
```

1.3.1.2 Git으로 프로젝트 관리하기

1.3.1.2.1 Github Flow Github가 Git으로 프로젝트를 관리하는 방법

1. main 줄기(branch)를 만들기
   실제로 사용자에게 배포(release)하는 줄기
   - 이 줄기에 다른 줄기를 합치려면, feature 줄기에서 합친후(merge), 테스트가 되어야한다.
2. `feature/`기능 줄기(branch)를 만들기
   새로운 기능을 추가하는 줄기
   - 새로운 기능이 필요할 때마다, 새로운 프랜치를 만든다.
     ex: `git branch feature/gotcha`

1.4 TODO

- [ ] Add Usage to documentation
- [ ] Add MySQL support

Chapter 2

클래스 색인

2.1 클래스 목록

다음은 클래스, 구조체, 공용체 그리고 인터페이스들입니다. (간략한 설명만을 보여줍니다) :

Chapter 3

파일 색인

3.1 파일 목록

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

Chapter 4

클래스 문서화

4.1 RESULT 구조체 참조

`#include <db.h>`

RESULT에 대한 협력 다이어그램:

Public 속성

- char name [30]
- int point
- int year
- int month
- int day
- int hour
- int min

4.1.1 상세한 설명

db.h 파일의 5 번째 라인에서 정의되었습니다.

4.1.2 멤버 데이터 문서화

4.1.2.1 day

```
int RESULT::day
```

db.h 파일의 10 번째 라인에서 정의되었습니다.

4.1.2.2 hour

```
int RESULT::hour
```

db.h 파일의 11 번째 라인에서 정의되었습니다.

4.1.2.3 min

```
int RESULT::min
```

db.h 파일의 12 번째 라인에서 정의되었습니다.

4.1.2.4 month

```
int RESULT::month
```

db.h 파일의 9 번째 라인에서 정의되었습니다.

4.1.2.5 name

```
char RESULT::name[30]
```

db.h 파일의 6 번째 라인에서 정의되었습니다.

4.1.2.6 point

```
int RESULT::point
```

db.h 파일의 7 번째 라인에서 정의되었습니다.

4.1.2.7 year

```
int RESULT::year
```

db.h 파일의 8 번째 라인에서 정의되었습니다.

이 구조체에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- include/db.h

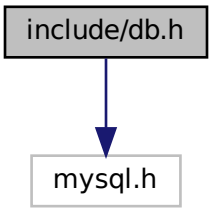
Chapter 5

파일 문서화

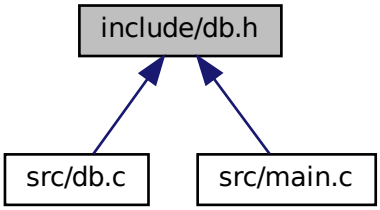
5.1 include/db.h 파일 참조

```
#include <mysql.h>
```

db.h에 대한 include 의존 그래프



이 그래프는 이 파일을 직/간접적으로 include 하는 파일들을 보여줍니다.:



클래스

- struct RESULT

함수

- void init_db ()
  MySQL DB를 초기화합니다
- void close_db ()
  MySQL DB를 종료합니다
- void read_db ()
  MySQL DB에서 데이터를 가져올 수 있습니다
- void write_db (int point)
  MySQL DB에 데이터를 저장할 수 있습니다
- int drop_table (MYSQL *conn)
  MySQL Table을 삭제합니다

5.1.1 함수 문서화

5.1.1.1 close_db()

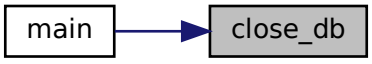
```
void close_db ( )
```

MySQL DB를 종료합니다

db.c 파일의 178 번째 라인에서 정의되었습니다.

```
178                 {
179   if (conn != NULL) {
180     mysql_close(conn);
181   }
182 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.1.1.2 drop_table()

```
int drop_table (
            MYSQL * conn )
```

MySQL Table을 삭제합니다

매개변수

conn	MySQL DB

db.c 파일의 124 번째 라인에서 정의되었습니다.

```
124                             {
125   printf("################\n");
126   printf("[drop_table] DROP TABLE log\n");
127   // drop table
128   if (mysql_query(conn, "DROP TABLE IF EXISTS log;")) {
129     printf("DROP TABLE Query failed\n");
130     return ERR;
131   }
132
133   return print_result(conn);
134 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:

5.1.1.3 init_db()

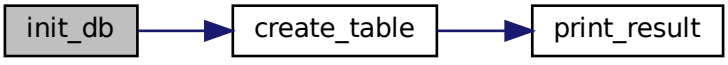
```
void init_db ( )
```

MySQL DB를 초기화합니다

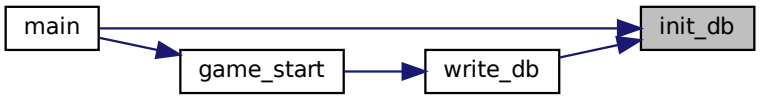
db.c 파일의 161 번째 라인에서 정의되었습니다.

```
161                 {
162   // establish connection with mysql
163   conn = mysql_init(NULL);
164   // if connection failed
165   if (!mysql_real_connect(conn, host, user, passwd, db, port, NULL, 0)) {
166     fprintf(stderr, "%s\n", mysql_error(conn));
167     mysql_close(conn);
168     exit(1);
169   }
170
171   // test_db(conn);
172   create_table(conn);
173 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.1.1.4 read_db()

```
void read_db ( )
```

MySQL DB에서 데이터를 가져올 수 있습니다

매개변수

name	이름을 통해 사용자 검색

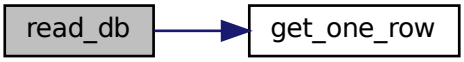
반환값

RESULT 구조체 주소 or NULL을 돌려줍니다

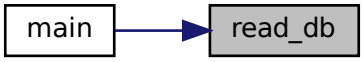
db.c 파일의 189 번째 라인에서 정의되었습니다.

```
189                 {
190   char query[256];
191   printf("이름을 입력하세요: ");
192   scanf("%s", tmp.name);
193   snprintf(query, sizeof(query), "SELECT * FROM log WHERE name = '%s'", tmp.name);
194   if (mysql_query(conn, query)) {
195     printf("SELECT Query failed\n");
196   }
197 
198   RESULT *res = get_one_row(conn);
199   if (res == NULL) {
200     system("clear");
201     printf("********************************\n");
202     printf("No History");
203     printf("********************************\n");
204   } else {
205       system("clear");
206       printf("********************************\n");
207       printf("%s\n", res->name);
208       printf("%d\n", res->point);
209       printf("time: %d-%d-%d %d:%d\n", res->year, res->month, res->day, res->hour, res->min);
210       printf("********************************\n");
211   }
212 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.1.1.5 write_db()

```
void write_db (
              int point )
```

MySQL DB에 데이터를 저장할 수 있습니다

매개변수

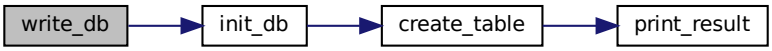
point	점수, 아이디

db.c 파일의 218 번째 라인에서 정의되었습니다.

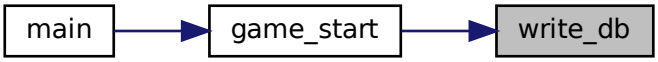
```
218                             {
219   printf("이름을 입력하세요: ");
220   scanf("%s", tmp.name);
221
222   new(point);
223   tmp.point = point;
224
225   time_t now_sec = time(NULL);
226   struct tm *now = localtime(&now_sec);
227
228   tmp.year = now->tm_year + 1900;
229   tmp.month = now->tm_mon + 1;
230   tmp.day = now->tm_mday;
231   tmp.hour = now->tm_hour;
232   tmp.min = now->tm_min;
233
234   if (conn == NULL) {
235     init_db();
236   }
237
238   char query[512];
```

```
239   // safe version of printf
240   // store query
241   snprintf(query, sizeof(query),
242            "INSERT INTO log (name, point, year, month, day, hour, min) VALUES "
243            "('%s', %d, %d, %d, %d, %d, %d)",
244            tmp.name, tmp.point, tmp.year, tmp.month, tmp.day, tmp.hour, tmp.min);
245   if (mysql_query(conn, query)) {
246     printf("INSERT Query failed: %s\n", mysql_error(conn));
247   }
248 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



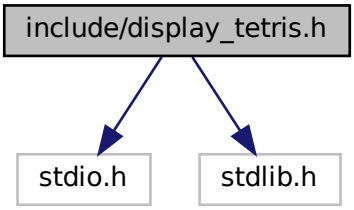
이 함수를 호출하는 함수들에 대한 그래프입니다.:



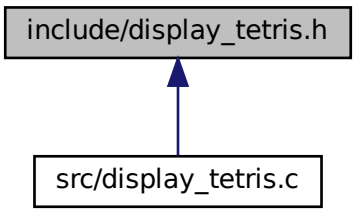
5.2 include/display_tetris.h 파일 참조

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

display_tetris.h에 대한 include 의존 그래프



이 그래프는 이 파일을 직/간접적으로 include 하는 파일들을 보여줍니다.:



매크로

- #define RESET_COLOR "\x1b[0m"
- #define RED_COLOR "\x1b[31m"
- #define GREEN_COLOR "\x1b[32m"
- #define YELLOW_COLOR "\x1b[33m"
- #define BLUE_COLOR "\x1b[34m"
- #define MAGENTA_COLOR "\x1b[35m"
- #define CYAN_COLOR "\x1b[36m"
- #define WHITE_COLOR "\x1b[37m"

변수

- int block [4][4][4]
- int tetris_table [21][10]
- int block_state

5.2.1 매크로 문서화

5.2.1.1 BLUE_COLOR

```
#define BLUE_COLOR "\x1b[34m"
```

display_tetris.h 파일의 11 번째 라인에서 정의되었습니다.

5.2.1.2 CYAN_COLOR

```
#define CYAN_COLOR "\x1b[36m"
```

display_tetris.h 파일의 13 번째 라인에서 정의되었습니다.

5.2.1.3 GREEN_COLOR

```
#define GREEN_COLOR "\x1b[32m"
```

display_tetris.h 파일의 9 번째 라인에서 정의되었습니다.

5.2.1.4 MAGENTA_COLOR

```
#define MAGENTA_COLOR "\x1b[35m"
```

display_tetris.h 파일의 12 번째 라인에서 정의되었습니다.

5.2.1.5 RED_COLOR

```
#define RED_COLOR "\x1b[31m"
```

display_tetris.h 파일의 8 번째 라인에서 정의되었습니다.

5.2.1.6 RESET_COLOR

```
#define RESET_COLOR "\x1b[0m"
```

display_tetris.h 파일의 7 번째 라인에서 정의되었습니다.

5.2.1.7 WHITE_COLOR

```
#define WHITE_COLOR "\x1b[37m"
```

display_tetris.h 파일의 14 번째 라인에서 정의되었습니다.

5.2.1.8 YELLOW_COLOR

```
#define YELLOW_COLOR "\x1b[33m"
```

display_tetris.h 파일의 10 번째 라인에서 정의되었습니다.

5.2.2 변수 문서화

5.2.2.1 block

```
int block[4][4][4]
```

초기값:

```
= {
    {{1, 1, 1, 1},
     {0, 0, 0, 0},
     {0, 0, 0, 0},
     {0, 0, 0, 0}},
    {{0, 0, 0, 1},
     {0, 0, 0, 1},
     {0, 0, 0, 1},
     {0, 0, 0, 1}},
    {{0, 0, 0, 0},
     {0, 0, 0, 0},
     {1, 1, 1, 1},
     {0, 0, 0, 0}},
    {{1, 0, 0, 0},
     {1, 0, 0, 0},
     {1, 0, 0, 0},
     {1, 0, 0, 0}}}
```

display_tetris.h 파일의 16 번째 라인에서 정의되었습니다.

5.2.2.2 block_state

```
int block_state  [extern]
```

update.c 파일의 7 번째 라인에서 정의되었습니다.

5.2.2.3 tetris_table

```
int tetris_table[21][10]
```

display_tetris.h 파일의 34 번째 라인에서 정의되었습니다.

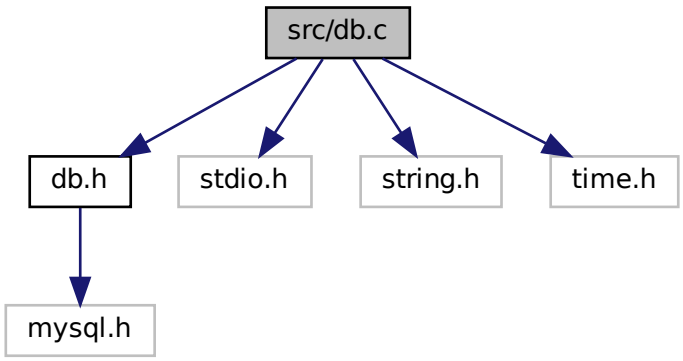
5.3 README.md 파일 참조

5.4 src/db.c 파일 참조

MySQL DB Functions

```
#include "db.h"
#include <stdio.h>
#include <string.h>
#include <time.h>
```

db.c에 대한 include 의존 그래프



매크로

- #define ERR 1
- #define OK 0

함수

- int print_result (MYSQL *conn)
  MySQL Result를 출력합니다
- RESULT * get_one_row (MYSQL *conn)
  MySQL DB에서 log 데이터를 가져올 수 있습니다
- int test_db (MYSQL *conn)
  MySQL DB를 테스트합니다
- int use_db (MYSQL *conn)
  MySQL DB를 사용합니다
- int drop_table (MYSQL *conn)
  MySQL Table을 삭제합니다
- int create_table (MYSQL *conn)
  MySQL DB를 생성합니다
- void init_db ()

MySQL DB를 초기화합니다

- void close_db ()
  MySQL DB를 종료합니다
- void read_db ()
  MySQL DB에서 데이터를 가져올 수 있습니다
- void write_db (int point)
  MySQL DB에 데이터를 저장할 수 있습니다

변수

- static MYSQL * conn = NULL
- static const char * host = "localhost"
- static const char * user = "root"
- static const char * passwd = "0000"
- static const char * db = "hm2j"
- static const int port = 3306
- static RESULT tmp

5.4.1 상세한 설명

MySQL DB Functions

작성자

xxxx

날짜

2024-03-21 @detail MySQL DB와 연결하고, 데이터를 저장, 가져올 수 있습니다.

5.4.2 매크로 문서화

5.4.2.1 ERR

```
#define ERR 1
```

db.c 파일의 12 번째 라인에서 정의되었습니다.

5.4.2.2 OK

```
#define OK 0
```

db.c 파일의 13 번째 라인에서 정의되었습니다.

5.4.3 함수 문서화

5.4.3.1 close_db()

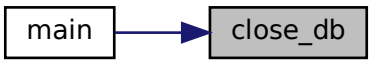
```
void close_db ( )
```

MySQL DB를 종료합니다

db.c 파일의 178 번째 라인에서 정의되었습니다.

```
178                 {
179   if (conn != NULL) {
180     mysql_close(conn);
181   }
182 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.4.3.2 create_table()

```
int create_table (
             MYSQL * conn )
```

MySQL DB를 생성합니다

매개변수

conn	MySQL DB

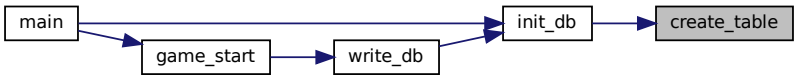
db.c 파일의 140 번째 라인에서 정의되었습니다.

```
140                             {
141   // create table
142   if (mysql_query(conn, "CREATE TABLE IF NOT EXISTS log ("
143                         "id INT AUTO_INCREMENT PRIMARY KEY,"
144                         "name VARCHAR(30),"
145                         "point INT,"
146                         "year INT,"
147                         "month INT,"
148                         "day INT,"
149                         "hour INT,"
150                         "min INT);")) {
151     printf("CREATE TABLE Query failed\n");
152     return ERR;
153   }
```

```
154
155   return print_result(conn);
156 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.4.3.3 drop_table()

```
int drop_table (
            MYSQL * conn )
```

MySQL Table을 삭제합니다

매개변수

conn	MySQL DB

db.c 파일의 124 번째 라인에서 정의되었습니다.

```
124                             {
125   printf("################\n");
126   printf("[drop_table] DROP TABLE log\n");
127   // drop table
128   if (mysql_query(conn, "DROP TABLE IF EXISTS log;")) {
129     printf("DROP TABLE Query failed\n");
130     return ERR;
131   }
132
133   return print_result(conn);
134 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:

5.4.3.4 get_one_row()

```
RESULT* get_one_row (
            MYSQL * conn )
```

MySQL DB에서 log 데이터를 가져올 수 있습니다

매개변수

conn	MySQL DB

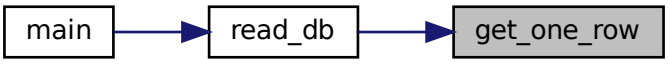
반환값

RESULT

db.c 파일의 48 번째 라인에서 정의되었습니다.

```
48                                      {
49   MYSQL_RES *res;
50   MYSQL_ROW row;
51   res = mysql_store_result(conn);
52   if (!res) {
53     fprintf(stderr, "%s\n", mysql_error(conn));
54     return NULL;
55   }
56
57   while ((row = mysql_fetch_row(res)) != NULL) {
58     // printf("%s, %s, %s, %s, %s, %s, %s\n", row[0], row[1], row[2],
59     // row[3],
60     //        row[4], row[5], row[6], row[7]);
61     // string copy
62     strcpy(tmp.name, row[1]);
63     // convert string to int
64     tmp.point = atoi(row[2]);
65     tmp.year = atoi(row[3]);
66     tmp.month = atoi(row[4]);
67     tmp.day = atoi(row[5]);
68     tmp.hour = atoi(row[6]);
69     tmp.min = atoi(row[7]);
70   }
71
72   return &tmp;
73 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.4.3.5 init_db()

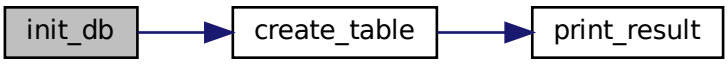
```
void init_db ( )
```

MySQL DB를 초기화합니다

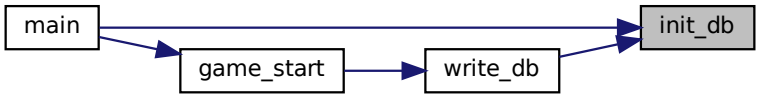
db.c 파일의 161 번째 라인에서 정의되었습니다.

```
161                 {
162   // establish connection with mysql
163   conn = mysql_init(NULL);
164   // if connection failed
165   if (!mysql_real_connect(conn, host, user, passwd, db, port, NULL, 0)) {
166     fprintf(stderr, "%s\n", mysql_error(conn));
167     mysql_close(conn);
168     exit(1);
169   }
170
171   // test_db(conn);
172   create_table(conn);
173 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.4.3.6 print_result()

```
int print_result (
            MYSQL * conn )
```

MySQL Result를 출력합니다

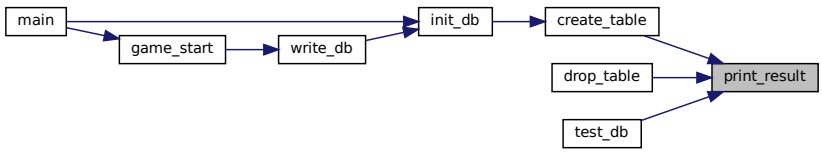
매개변수

conn	MySQL DB

db.c 파일의 28 번째 라인에서 정의되었습니다.

```
28                                  {
29   MYSQL_RES *res;
30   MYSQL_ROW row;
31   res = mysql_store_result(conn);
32   if (!res) {
33     fprintf(stderr, "%s\n", mysql_error(conn));
34     return ERR;
35   }
36   while ((row = mysql_fetch_row(res)) != NULL) {
37     printf("%s\n", row[0]);
38   }
39
40   return OK;
41 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.4.3.7 read_db()

```
void read_db ( )
```

MySQL DB에서 데이터를 가져올 수 있습니다

매개변수

name	이름을 통해 사용자 검색

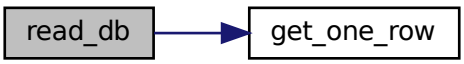
반환값

RESULT 구조체 주소 or NULL을 돌려줍니다

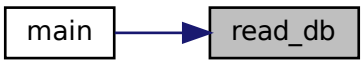
db.c 파일의 189 번째 라인에서 정의되었습니다.

```
189                 {
190   char query[256];
191   printf("이름을 입력하세요: ");
192   scanf("%s", tmp.name);
193   snprintf(query, sizeof(query), "SELECT * FROM log WHERE name = '%s'", tmp.name);
194   if (mysql_query(conn, query)) {
195     printf("SELECT Query failed\n");
196   }
197
198   RESULT *res = get_one_row(conn);
199   if (res == NULL) {
200     system("clear");
201     printf("********************************\n");
202     printf("No History");
203     printf("********************************\n");
204   } else {
205       system("clear");
206       printf("********************************\n");
207       printf("%s\n", res->name);
208       printf("%d\n", res->point);
209       printf("time: %d-%d-%d %d:%d\n", res->year, res->month, res->day, res->hour, res->min);
210       printf("********************************\n");
211   }
212 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.4.3.8 test_db()

```
int test_db (
              MYSQL * conn )
```

MySQL DB를 테스트합니다

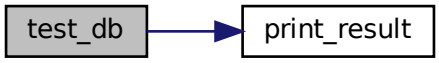
매개변수

conn	MySQL DB

db.c 파일의 79 번째 라인에서 정의되었습니다.

```
79                          {
80   printf("################\n");
81   printf("[test_db] SHOW databases\n");
82
83   // show all databases
84   if (mysql_query(conn, "SHOW DATABASES;")) {
85     printf("SHOW Query failed\n");
86     return ERR;
87   }
88
89   if (print_result(conn)) {
90     printf("Print Result failed\n");
91     return ERR;
92   };
93
94   printf("################\n");
95   printf("[show_table] SHOW TABLES\n");
96   if (mysql_query(conn, "SHOW TABLES;")) {
97     printf("SHOW TABLES Query failed\n");
98     return ERR;
99   }
100
101   return print_result(conn);
102 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



5.4.3.9 use_db()

```
int use_db (
              MYSQL * conn )
```

MySQL DB를 사용합니다

매개변수

conn	MySQL DB

db.c 파일의 108 번째 라인에서 정의되었습니다.

```
108                         {
109   printf("################\n");
110   printf("[use_db] USE hm2j\n");
111   // use database
112   if (mysql_query(conn, "USE hm2j")) {
113     printf("USE Query failed\n");
114     return ERR;
115   }
116
117   return OK;
118 }
```

5.4.3.10 write_db()

```
void write_db (
            int point )
```

MySQL DB에 데이터를 저장할 수 있습니다

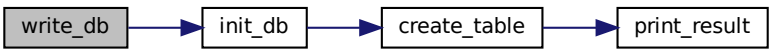
매개변수

point	점수, 아이디

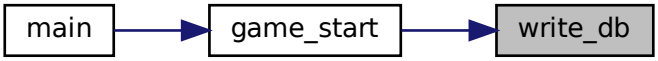
db.c 파일의 218 번째 라인에서 정의되었습니다.

```
218                         {
219   printf("이름을 입력하세요: ");
220   scanf("%s", tmp.name);
221
222   new(point);
223   tmp.point = point;
224
225   time_t now_sec = time(NULL);
226   struct tm *now = localtime(&now_sec);
227
228   tmp.year = now->tm_year + 1900;
229   tmp.month = now->tm_mon + 1;
230   tmp.day = now->tm_mday;
231   tmp.hour = now->tm_hour;
232   tmp.min = now->tm_min;
233
234   if (conn == NULL) {
235     init_db();
236   }
237
238   char query[512];
239   // safe version of printf
240   // store query
241   snprintf(query, sizeof(query),
242            "INSERT INTO log (name, point, year, month, day, hour, min) VALUES "
243            "('%s', %d, %d, %d, %d, %d, %d)",
244            tmp.name, tmp.point, tmp.year, tmp.month, tmp.day, tmp.hour, tmp.min);
245   if (mysql_query(conn, query)) {
246     printf("INSERT Query failed: %s\n", mysql_error(conn));
247   }
248 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.4.4 변수 문서화

5.4.4.1 conn

```
MYSQL* conn = NULL  [static]
```

db.c 파일의 15 번째 라인에서 정의되었습니다.

5.4.4.2 db

```
const char* db = "hm2j"  [static]
```

db.c 파일의 19 번째 라인에서 정의되었습니다.

5.4.4.3 host

```
const char* host = "localhost"  [static]
```

db.c 파일의 16 번째 라인에서 정의되었습니다.

5.4.4.4 passwd

```
const char* passwd = "0000"  [static]
```

db.c 파일의 18 번째 라인에서 정의되었습니다.

5.4.4.5 port

```
const int port = 3306  [static]
```

db.c 파일의 20 번째 라인에서 정의되었습니다.

5.4.4.6 tmp

```
RESULT tmp  [static]
```

db.c 파일의 22 번째 라인에서 정의되었습니다.

5.4.4.7 user

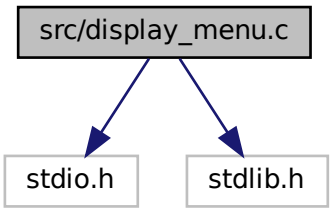
```
const char* user = "root"  [static]
```

db.c 파일의 17 번째 라인에서 정의되었습니다.

5.5 src/display_menu.c 파일 참조

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

display_menu.c에 대한 include 의존 그래프



함수

- int display_menu ()

5.5.1 함수 문서화

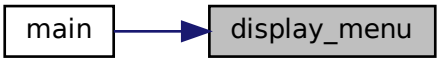
5.5.1.1 display_menu()

```
int display_menu ( )
```

display_menu.c 파일의 4 번째 라인에서 정의되었습니다.

```
5 {
6     int menu;
7
8     system("clear");
9     printf("\t\t\t _ _ _ _ _ _\n\t\t\t/ \_ _ _ _ \\n\t\t\t|¯\∩·ω·) \\n\t\t\t|    |¯¯∪¯¯|\n");
10     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   TEAM HM2J_c's TETRIS
       GAME\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n\t\t          \\ (••) /\n\t\t           \\     /\n");
11
12     printf("\t\t-----------------------------\n");
13     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n   \t\t\t  GAME MENU\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
14     printf("\t\t-----------------------------\n");
15
16     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   1. Game Start\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
17     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   2. Search history\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
18     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   3. Record Output\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
19     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   4. Quit\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
20
21     printf("\n");
22
23     // printf("\t\t      ┌─┐ ┌─┐ ┌─┐ ┌┐ ┌┐ \n\t\t      └┐ | | ┌┐ | └┐ | | | | |\n\t\t
       ┌┘ | | | | ┌┘ | | └┘ |\n\t\t      | ┌┘ | | | | ┌┘ └┐ |\n\t\t      | └┐ | └┘ | | └┐
       | |\n\t\t      └─┘ └─┘ └─┘   └┘ \n");
24
25     printf("Select The Number: ");
26     scanf("%d", &menu);
27     return menu;
28 }
```

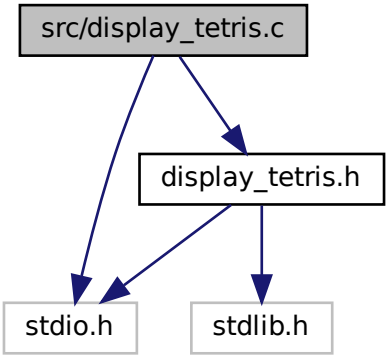
이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.6 src/display_tetris.c 파일 참조

```
#include <stdio.h>
#include "display_tetris.h"
```

display_tetris.c에 대한 include 의존 그래프



함수

- void display_tetris ()

변수

- int i_block [4][4][4]
- int t_block [4][4][4]
- int s_block [4][4][4]
- int z_block [4][4][4]
- int l_block [4][4][4]
- int j_block [4][4][4]
- int o_block [4][4][4]
- int next_block_number

5.6.1 함수 문서화

5.6.1.1 display_tetris()

```
void display_tetris ( )
```

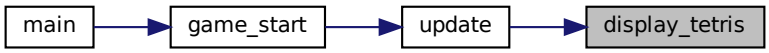
display_tetris.c 파일의 13 번째 라인에서 정의되었습니다.

```
14 {
15     int i, j;
16     int(*block_pointer)[4][4][4] = NULL;
17     int block_color; // 인트 블럭 색상 선언
18 
19     switch (next_block_number)
20     {
21     case 0:
22         block_pointer = &i_block;
```

```
23          break;
24      case 1:
25          block_pointer = &t_block;
26          break;
27      case 2:
28          block_pointer = &s_block;
29          break;
30      case 3:
31          block_pointer = &z_block;
32          break;
33      case 4:
34          block_pointer = &l_block;
35          break;
36      case 5:
37          block_pointer = &j_block;
38          break;
39      case 6:
40          block_pointer = &o_block;
41          break;
42      }
43
44      // 터미널 화면을 지우고 다음 블록 출력
45      system("clear");
46      printf("Next Block\n");
47
48      //블록 출력 시작
49      for (i = 0; i < 4; i++)
50      {
51          for (j = 0; j < 4; j++)
52          {
53              // 블록이 있는 경우 해당 색상으로 출력합니다.
54              if ((*block_pointer)[0][i][j] == 1)
55              {
56                  printf(RED_COLOR "■ " RESET_COLOR);
57              }else if ((*block_pointer)[0][i][j] == 2){
58                  printf(GREEN_COLOR "■ " RESET_COLOR);
59              }else if ((*block_pointer)[0][i][j] == 3){
60                  printf(YELLOW_COLOR "■ " RESET_COLOR);
61              }else if ((*block_pointer)[0][i][j] == 4){
62                  printf(BLUE_COLOR "■ " RESET_COLOR);
63              }else if ((*block_pointer)[0][i][j] == 5){
64                  printf(MAGENTA_COLOR "■ " RESET_COLOR);
65              }else if ((*block_pointer)[0][i][j] == 6){
66                  printf(CYAN_COLOR "■ " RESET_COLOR);
67              }else if ((*block_pointer)[0][i][j] == 7){
68                  printf(WHITE_COLOR "■ " RESET_COLOR);
69              }else {
70                  printf("□ ");
71              }
72          }
73          printf("\n");
74      }
75      // 블록이 없는 경우 공백을 출력합니다.
76      printf("\n\n"); //두줄 공백 출력하는 것
77      printf("Tetris Table\n");
78      printf("\t┌┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┐\n\t┌ ┬ ┬ ┬ ┬ ┬ ┐\n\t| | | ^ ^ | | 살려
        줘!!\n\t| | | ( ≧ Д ≦ ) | \n\t| | | Φ Φ | |\n\t└ ┴ ┴ ┴ ┴ ┴ ┘
        \n\t└┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┘\n");
79      for (i = 2; i < 21; i++)
80      {
81          printf("\t");
82          for (j = 0; j < 10; j++)
83          {
84              if (j == 0 || j == 9)
85              {
86                  printf("= ");
87              }
88              else if (i == 20 && (j > 0 && j < 9))
89              {
90                  printf("=");
91              }
92              else if (tetris_table[i][j] == 1){      //블럭포인터로 표현했었었는데, 테트리스테이블로 다시 함.
93                  printf(RED_COLOR "■ " RESET_COLOR);
94              }else if (tetris_table[i][j] == 2){
95                  printf(GREEN_COLOR "■ " RESET_COLOR);
96              }else if (tetris_table[i][j] == 3){
97                  printf(YELLOW_COLOR "■ " RESET_COLOR);
98              }else if (tetris_table[i][j] == 4){
99                  printf(BLUE_COLOR "■ " RESET_COLOR);
100             }else if (tetris_table[i][j] == 5){
101                 printf(MAGENTA_COLOR "■ " RESET_COLOR);
102             }else if (tetris_table[i][j] == 6){
103                 printf(CYAN_COLOR "■ " RESET_COLOR);
104             }else if (tetris_table[i][j] == 7){
105                 printf(WHITE_COLOR "■ " RESET_COLOR);
106             }else { // 블록이 없는 경우 공백
107                 printf("  ");
```

```
108             }
109         }
110         printf("\n");
111     }
112     printf("\t 신해철 - 그대에게\n\t0:35 ———————— -3:48\n\t        ||        \n");
113 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.6.2 변수 문서화

5.6.2.1 i_block

```
int i_block[4][4][4]  [extern]
```

update.c 파일의 17 번째 라인에서 정의되었습니다.

5.6.2.2 j_block

```
int j_block[4][4][4]  [extern]
```

update.c 파일의 122 번째 라인에서 정의되었습니다.

5.6.2.3 l_block

```
int l_block[4][4][4]  [extern]
```

update.c 파일의 101 번째 라인에서 정의되었습니다.

5.6.2.4 next_block_number

```
int next_block_number  [extern]
```

update.c 파일의 12 번째 라인에서 정의되었습니다.

5.6.2.5 o_block

```
int o_block[4][4][4]  [extern]
```

update.c 파일의 143 번째 라인에서 정의되었습니다.

5.6.2.6 s_block

```
int s_block[4][4][4]  [extern]
```

update.c 파일의 59 번째 라인에서 정의되었습니다.

5.6.2.7 t_block

```
int t_block[4][4][4]  [extern]
```

update.c 파일의 38 번째 라인에서 정의되었습니다.

5.6.2.8 z_block

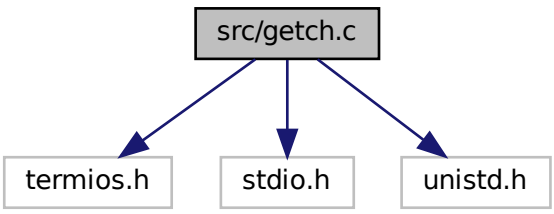
```
int z_block[4][4][4]  [extern]
```

update.c 파일의 80 번째 라인에서 정의되었습니다.

5.7 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)

5.7.1 매크로 문서화

5.7.1.1 CCHAR

```
#define CCHAR 0
```

getch.c 파일의 6 번째 라인에서 정의되었습니다.

5.7.1.2 CTIME

```
#define CTIME 1
```

getch.c 파일의 10 번째 라인에서 정의되었습니다.

5.7.1.3 FLAG

```
#define FLAG TCSAFLUSH
```

5.7.2 함수 문서화

5.7.2.1 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 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:

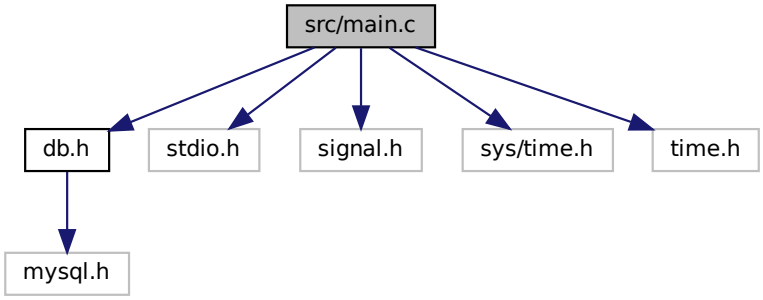
5.8 src/main.c 파일 참조

테트리스 게임을 실행하는 메인 함수

```
#include "db.h"
#include <stdio.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

함수

- int display_menu ()
- void display_tetris ()
- int update (int signum)
- void save_result (int)
- int print_result (void)
- int print_color ()
- int game_start ()

  이 함수는 signal을 설정하는 함수입니다.
- int main ()
- int init_tetris_table ()

변수

- int game
- int x = 3
- int y = 0
- int point = 0
- int tetris_table [21][10]
- char q

5.8.1 상세한 설명

테트리스 게임을 실행하는 메인 함수

작성자

최수길

날짜

2024-3-20

이 게임은 signal을 사용해서 1/60 초 마다 화면을 갱신 하는 것을 이용한 프로그램입니다.

5.8.2 매크로 문서화

5.8.2.1 GAME_END

```
#define GAME_END 1
```

main.c 파일의 17 번째 라인에서 정의되었습니다.

5.8.2.2 GAME_START

```
#define GAME_START 0
```

main.c 파일의 16 번째 라인에서 정의되었습니다.

5.8.3 함수 문서화

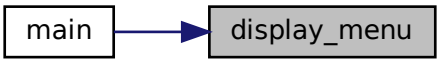
5.8.3.1 display_menu()

```
int display_menu ( )
```

display_menu.c 파일의 4 번째 라인에서 정의되었습니다.

```
5 {
6     int menu;
7
8     system("clear");
9     printf("\t\t\t__ _ _ _ _ _\n\t\t\t/ \_ _ _ _ \\n\t\t\t|¯\∩·ω·) \\n\t\t\t|    |¯¯∪¯¯|\n");
10     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   TEAM HM2J_c's TETRIS
       GAME\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n\t\t          \\ (••) /\n\t\t            \\     /\n");
11
12     printf("\t\t-----------------------------\n");
13     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n   \t\t\t  GAME MENU\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
14     printf("\t\t-----------------------------\n");
15
16     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   1. Game Start\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
17     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   2. Search history\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
18     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   3. Record Output\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
19     printf("\t\t|¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯|\n\t\t   4. Quit\n\t\t|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _|\n");
20
21     printf("\n");
22
23     // printf("\t\t      ┌─┐ ┌─┐ ┌─┐ ┌┐ ┌┐ \n\t\t      └┐ | | ┌┐ | └┐ | | | | |\n\t\t
       ┌─┘ | | | | ┌─┘ | | └┘ |\n\t\t      | ┌─┘ | | | | ┌─┘ └┐ |\n\t\t      | └┐ | └┘ | | └┐
       | | \n\t\t      └─┘ └─┘ └─┘   └┘ \n");
24
25     printf("Select The Number: ");
26     scanf("%d", &menu);
27     return menu;
28 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.8.3.2 display_tetris()

```
void display_tetris ( )
```

display_tetris.c 파일의 13 번째 라인에서 정의되었습니다.

```
14 {
15     int i, j;
16     int(*block_pointer)[4][4][4] = NULL;
17     int block_color; // 인트 블럭 색상 선언
18
19     switch (next_block_number)
20     {
21     case 0:
22         block_pointer = &i_block;
23         break;
24     case 1:
25         block_pointer = &t_block;
26         break;
27     case 2:
28         block_pointer = &s_block;
29         break;
30     case 3:
31         block_pointer = &z_block;
```

```
32          break;
33      case 4:
34          block_pointer = &l_block;
35          break;
36      case 5:
37          block_pointer = &j_block;
38          break;
39      case 6:
40          block_pointer = &o_block;
41          break;
42      }
43
44      // 터미널 화면을 지우고 다음 블록 출력
45      system("clear");
46      printf("Next Block\n");
47
48      //블록 출력 시작
49      for (i = 0; i < 4; i++)
50      {
51          for (j = 0; j < 4; j++)
52          {
53              // 블록이 있는 경우 해당 색상으로 출력합니다.
54              if ((*block_pointer)[0][i][j] == 1)
55              {
56                  printf(RED_COLOR "■ " RESET_COLOR);
57              }else if ((*block_pointer)[0][i][j] == 2){
58                  printf(GREEN_COLOR "■ " RESET_COLOR);
59              }else if ((*block_pointer)[0][i][j] == 3){
60                  printf(YELLOW_COLOR "■ " RESET_COLOR);
61              }else if ((*block_pointer)[0][i][j] == 4){
62                  printf(BLUE_COLOR "■ " RESET_COLOR);
63              }else if ((*block_pointer)[0][i][j] == 5){
64                  printf(MAGENTA_COLOR "■ " RESET_COLOR);
65              }else if ((*block_pointer)[0][i][j] == 6){
66                  printf(CYAN_COLOR "■ " RESET_COLOR);
67              }else if ((*block_pointer)[0][i][j] == 7){
68                  printf(WHITE_COLOR "■ " RESET_COLOR);
69              }else {
70                  printf("□ ");
71              }
72          }
73          printf("\n");
74      }
75      // 블록이 없는 경우 공백을 출력합니다.
76      printf("\n\n"); //두줄 공백 출력하는 것
77      printf("Tetris Table\n");
78      printf("\t┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬\n\t┌ ┬ ┬ ┬ ┬ ┬ ┐\n\t|  |  |  ^  ^  |  |    살려
        줘!!\n\t|  |  |  ( ≧ Д ≦ )  |  \n\t|  |  |  ϕ  ϕ  |  |\n\t└ ┴ ┴ ┴ ┴ ┴ ┘
        \n\t┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴\n");
79      for (i = 2; i < 21; i++)
80      {
81          printf("\t");
82          for (j = 0; j < 10; j++)
83          {
84              if (j == 0 || j == 9)
85              {
86                  printf("= ");
87              }
88              else if (i == 20 && (j > 0 && j < 9))
89              {
90                  printf("=");
91              }
92              else if (tetris_table[i][j] == 1){      //블럭포인터로 표현했었었는데, 테트리스테이블로 다시 함.
93                  printf(RED_COLOR "■ " RESET_COLOR);
94              }else if (tetris_table[i][j] == 2){
95                  printf(GREEN_COLOR "■ " RESET_COLOR);
96              }else if (tetris_table[i][j] == 3){
97                  printf(YELLOW_COLOR "■ " RESET_COLOR);
98              }else if (tetris_table[i][j] == 4){
99                  printf(BLUE_COLOR "■ " RESET_COLOR);
100             }else if (tetris_table[i][j] == 5){
101                 printf(MAGENTA_COLOR "■ " RESET_COLOR);
102             }else if (tetris_table[i][j] == 6){
103                 printf(CYAN_COLOR "■ " RESET_COLOR);
104             }else if (tetris_table[i][j] == 7){
105                 printf(WHITE_COLOR "■ " RESET_COLOR);
106             }else { // 블록이 없는 경우 공백
107                 printf("  ");
108             }
109         }
110         printf("\n");
111     }
112     printf("\t 신해철 - 그대에게\n\t0:35 ──────────── -3:48\n\t      ||      \n");
113 }
```

5.8.3.3 game_start()

```
int game_start ( )
```

이 함수는 signal을 설정하는 함수입니다.

매개변수

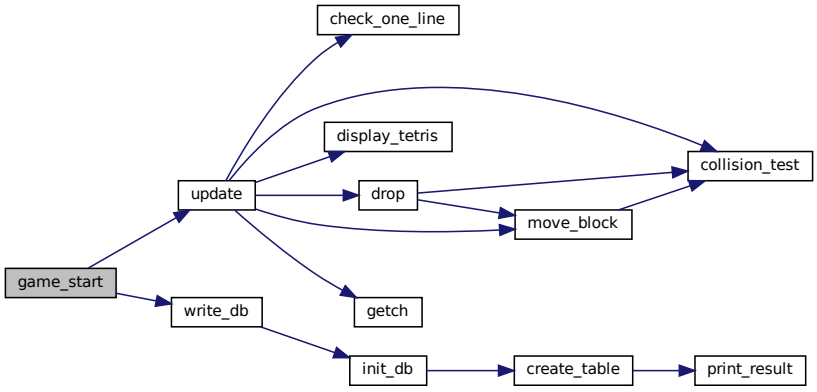
signum	signal 에 연결된 함수에 전달되는 인자
timer	signal을 설정하는 구조체 여기에 갱신 시간을 정한다.

이 함수에서는 x, y, point 값을 초기화 시킨다.

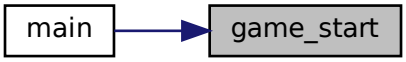
main.c 파일의 38 번째 라인에서 정의되었습니다.

```
39 {
40     // signal 설정
41     static struct itimerval timer;
42     signal(SIGVTALRM, update);
43     timer.it_value.tv_sec = 0;
44     timer.it_value.tv_usec = 16667;
45     timer.it_interval.tv_sec = 0;
46     timer.it_interval.tv_usec = 16667;
47     setitimer(ITIMER_VIRTUAL, &timer, NULL);
48     while (1)
49     {
50         if (game == GAME_END)
51         {
52             signal(SIGVTALRM, SIG_IGN);
53             // save_result(point);
54             write_db(point);
55             x = 3;
56             y = 0;
57             point = 0;
58             return 1;
59         }
60     }
61     game = GAME_END;
62 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.8.3.4 init_tetris_table()

```
int init_tetris_table ( )
```

main.c 파일의 92 번째 라인에서 정의되었습니다.

```
93 {
94     int i, j;
95     // 모든 데이터에 0을 넣는다.
96     for (i = 0; i < 21; i++)
97     {
98         for (j = 0; j < 10; j++)
99         {
100             tetris_table[i][j] = 0;
101         }
102     }
103     // 맨 아래칸에 1 추가 (바닥)
104     for (j = 1; j < 9; j++)
105     {
106         tetris_table[20][j] = 1;
107     }
108     // 양 옆에 1 추가 (벽)
109     for (i = 0; i < 21; i++)
110     {
111         tetris_table[i][0] = 1;
112         tetris_table[i][9] = 1;
113     }
114 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:

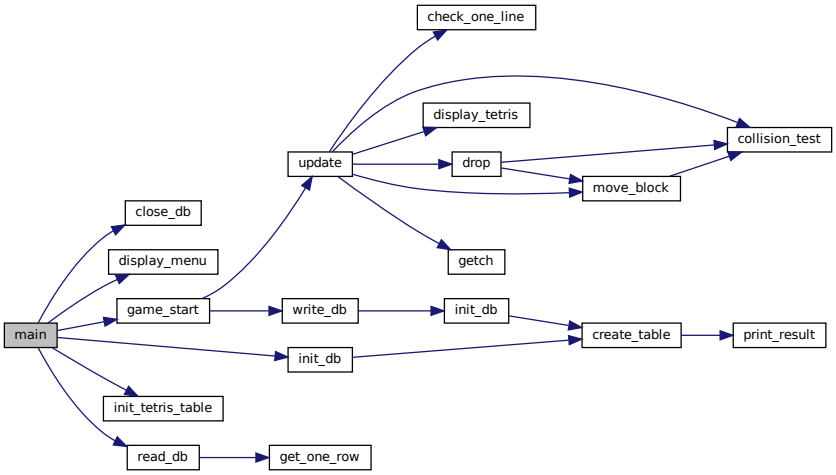
5.8.3.5 main()

```
int main ( )
```

main.c 파일의 64 번째 라인에서 정의되었습니다.

```
65 {
66     int menu = 1;
67     menu = display_menu();
68     switch (menu)
69     {
70     case 1:
71         game = GAME_START;
72         init_tetris_table();
73         menu = game_start();
74         break;
75     case 2:
76         printf("Search history\n");
77         init_db();
78         read_db();
79         close_db();
80         break;
81     case 3:
82         printf("Record Output\n");
83         // print_result();
84         break;
85     case 4:
86         printf("Quit\n");
87         break;
88     }
89     return 0;
90 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



5.8.3.6 print_color()

```
int print_color ( )
```

5.8.3.7 print_result()

```
int print_result (
            void  )
```

5.8.3.8 save_result()

```
void save_result (
            int  )
```

5.8.3.9 update()

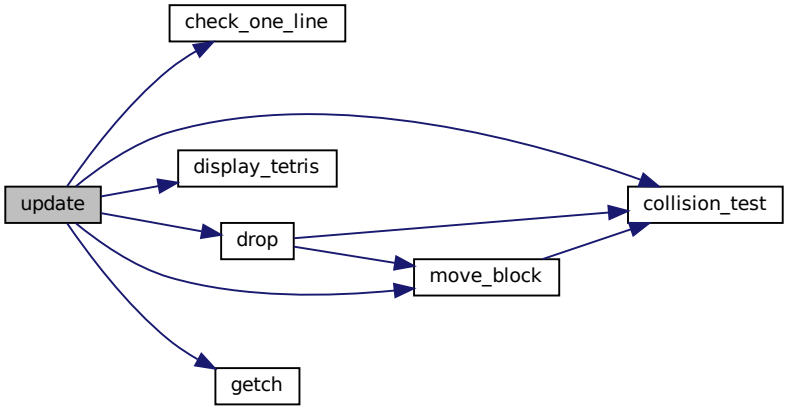
```
int update (
            int signum )
```

update.c 파일의 174 번째 라인에서 정의되었습니다.

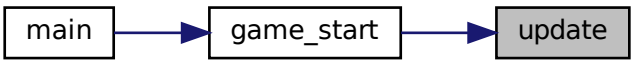
```
175 {
176     static int downcount = 0;
177     static int setcount = 0;
178     static long speedcount = 0;
179     static int countrange = 5;
180     static int firststart = 0;
181 
182     char ch;
183 
184     srand((unsigned)time(NULL));
185 
186     if (firststart == 0)
187     {
188         block_number = rand() % 7;
189         if (firststart == 0)
190             firststart++;
191     }
192 
193     display_tetris();
194     check_one_line();
195 
196     if (downcount == countrange - 1)
197     {
198         point += 1;
199         move_block(DOWN);
200     }
201 
202     if (speedcount == 499)
203     {
204         if (countrange != 1)
205             countrange--;
206     }
207 
208     downcount++;
209     downcount %= countrange;
210     speedcount++;
211     speedcount %= 500;
212 
213     if (x == 3 && y == 0)
214     {
215         if (collision_test(LEFT) || collision_test(RIGHT) || collision_test(DOWN) ||
      collision_test(ROTATE))
216         {
217             printf("\n Game End! \n");
218             downcount = 0;
219             setcount = 0;
220             speedcount = 0;
221             countrange = 5;
222             firststart = 0;
223             game = GAME_END;
224         }
225     }
```

```
226
227     if (collision_test(DOWN))
228     {
229         if (setcount == 9)
230         {
231             block_number = next_block_number;
232             next_block_number = rand() % 7;
233             block_state = 0;
234             x = 3;
235             y = 0;
236         }
237         setcount++;
238         setcount %= 10;
239     }
240
241     ch = getch();
242
243     switch (ch)
244     {
245     case 74:
246     case 106:
247         move_block(LEFT);
248         break;
249     case 76:
250     case 108:
251         move_block(RIGHT);
252         break;
253     case 75:
254     case 107:
255         move_block(DOWN);
256         break;
257     case 73:
258     case 105:
259         move_block(ROTATE);
260         break;
261     case 65:
262     case 97:
263         drop();
264         break;
265     case 80:
266     case 112:
267         downcount = 0;
268         setcount = 0;
269         speedcount = 0;
270         countrange = 5;
271         firststart = 0;
272         game = GAME_END;
273         break;
274     default:
275         break;
276     }
277     return 0;
278 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.8.4 변수 문서화

5.8.4.1 game

```
int game
```

main.c 파일의 21 번째 라인에서 정의되었습니다.

5.8.4.2 point

```
int point = 0
```

main.c 파일의 25 번째 라인에서 정의되었습니다.

5.8.4.3 q

```
char q
```

main.c 파일의 30 번째 라인에서 정의되었습니다.

5.8.4.4 tetris_table

```
int tetris_table[21][10]  [extern]
```

display_tetris.h 파일의 34 번째 라인에서 정의되었습니다.

5.8.4.5 x

```
int x = 3
```

main.c 파일의 23 번째 라인에서 정의되었습니다.

5.8.4.6 y

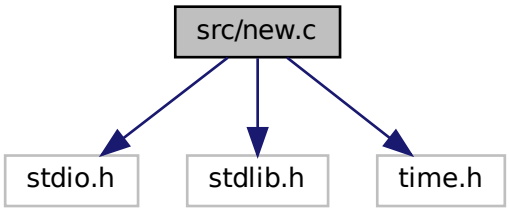
```
int y = 0
```

main.c 파일의 24 번째 라인에서 정의되었습니다.

5.9 src/new.c 파일 참조

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

new.c에 대한 include 의존 그래프



함수

- int getRandomScore ()
- int chooseDoll (int score)
- int new (int a)

5.9.1 함수 문서화

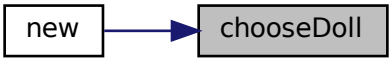
5.9.1.1 chooseDoll()

```
int chooseDoll (
            int score )
```

new.c 파일의 25 번째 라인에서 정의되었습니다.

```
25                          {
26     if (score >= 200000) {
27         printf(".    ∧_∧\n");
28
29         printf("  (  つ┬⊃\n");
30         printf("ε (_)へ^\n");
31         printf("(    (  ･o･)\n");
32         printf("◎--◎    ⊃   ⊃\n");
33         return 10; // 200,000 이상일 경우 10번 인형 획득
34     } else if (score >= 100000) {
35         printf("      ∧„∧\n");
36         printf("☆(ω･*\n");
37         printf("     (   ⊂)\n");
38         printf("      し―Ｊ\n");
39
40         return 9; // 100000 이상일 경우 9번 인형 획득
41     } else if (score >= 50000) {
42         printf(".   ∧„∧\n");
43         printf(" ∩(^ 0 ^)∩\n");
44         printf("          ノ\n");
45         printf("  . |     |\n");
46         printf("    U ^ U.\n");
47
48         return 8; // 50000 이상일 경우 8번 인형 획득
49     } else if (score >= 30000) {
50         printf(". ♪  ∧,_∧\n");
51         printf("    (´ ω ｀) ))\n");
52         printf("   (( (  つ  、 ♪\n");
53         printf("     〉とノ  )))\n");
54         printf("    (__ノ^(_)\n");
55         return 7; // 30000 이상일 경우 7번 인형 획득
56     } else if (score >= 20000) {
57
58         printf("(    ()  )\n");
59         printf("/    ^\n");
60         printf("人_＿つ_つ\n");
61         return 6; // 20000 이상일 경우 6번 인형 획득
62     } else if (score >= 15000) {
63         printf(".  ∧∧  ■");
64         printf("  (^ω^)ノ  ");
65         printf("  く    /");
66         printf("    ∪∪");
67
68         return 5; // 15000 이상일 경우 5번 인형 획득
69     } else if (score >= 10000) {
70
71
72         return 4; // 10000 이상일 경우 4번 인형 획득
73     } else if (score >= 5000) {
74
75
76         printf("       (\(__u_u)\n");
77         return 3; // 5000 이상일 경우 3번 인형 획득
78     } else if (score >= 3000) {
79         printf(". /) /)\n");
80
81         return 2; // 3000 이상일 경우 2번 인형 획득
82     } else if (score >= 1000) {
83         printf(".∩    ∩\n");
84
85         return 1; // 1000 이상일 경우 1번 인형 획득
86     }
87 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.9.1.2 getRandomScore()

```
int getRandomScore ( )
```

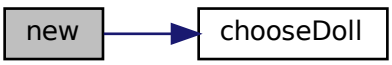
5.9.1.3 new()

```
int new (
              int a )
```

new.c 파일의 9 번째 라인에서 정의되었습니다.

```
9                {
10
11      int score = a;
12      int doll = chooseDoll(score); // 점수에 따라 인형 선택
13      printf("당신의 점수는 ... %d\n", score);
14      if (score < 100) {
15          printf(".   \\n  \  \\n    \  ¯¯¯\  ))\n     \_つ))\n  /  /          „\n|  o●          ●o
        |\n、: (_人_)  :。ノ\n");
16          printf("이것이 당신의 실력..? 더 커서 돌아오세요.\n");
17      } else {
18          printf("축하합니다! %d번 인형을 획득하셨습니다!\n", doll);
19      }
20
21      return 0;
22  }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:

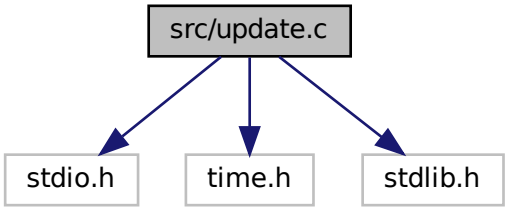


5.10 src/result.c 파일 참조

5.11 src/update.c 파일 참조

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

update.c에 대한 include 의존 그래프



매크로

- #define GAME_START 0
- #define GAME_END 1

열거형 타입

- enum COMMAND { DOWN , LEFT , RIGHT , ROTATE }

함수

- char getch ()
- int display_tetris ()
- int update (int signum)
- int move_block (int command)
- int collision_test (int command)
- int drop (void)
- int check_one_line (void)

변수

- int x
- int y
- int block_state = 0
- int block [4][4][4]
- int point
- int tetris_table [21][10]
- int block_number
- int next_block_number
- int game
- int i_block [4][4][4]
- int t_block [4][4][4]
- int s_block [4][4][4]
- int z_block [4][4][4]
- int l_block [4][4][4]
- int j_block [4][4][4]
- int o_block [4][4][4]

5.11.1 매크로 문서화

5.11.1.1 GAME_END

```
#define GAME_END 1
```

update.c 파일의 165 번째 라인에서 정의되었습니다.

5.11.1.2 GAME_START

```
#define GAME_START 0
```

update.c 파일의 164 번째 라인에서 정의되었습니다.

5.11.2 열거형 타입 문서화

5.11.2.1 COMMAND

```
enum COMMAND
```

열거형 멤버

DOWN	
LEFT	
RIGHT	
ROTATE	

update.c 파일의 167 번째 라인에서 정의되었습니다.

```
168 {
169     DOWN,
170     LEFT,
171     RIGHT,
172     ROTATE
173 } COMMAND;
```

5.11.3 함수 문서화

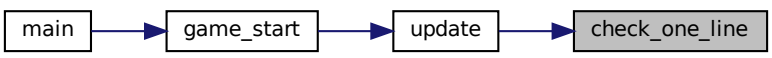
5.11.3.1 check_one_line()

```
int check_one_line (
              void  )
```

update.c 파일의 468 번째 라인에서 정의되었습니다.

```
469 {
470     int i, j;
471     int ti, tj;
472     int line_hole;
473 
474     for (i = 19; i > 0; i--)
475     {
476         line_hole = 0;
477         for (j = 1; j < 9; j++)
478         {
479             if (tetris_table[i][j] == 0)
480             {
481                 line_hole = 1;
482             }
483         }
484 
485         if (line_hole == 0)
486         {
487             point += 1000;
488             for (ti = i; ti > 0; ti--)
489             {
490                 for (tj = 0; tj < 9; tj++)
491                 {
492                     tetris_table[ti][tj] = tetris_table[ti - 1][tj];
493                 }
494             }
495         }
496     }
497 
498     return 0;
499 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.11.3.2 collision_test()

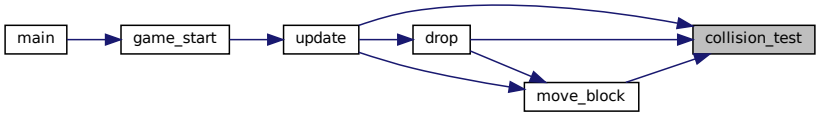
```
int collision_test (
            int command )
```

update.c 파일의 370 번째 라인에서 정의되었습니다.

```
371 {
372     int i, j;
373     int tempx, tempy;
374     int oldx, oldy;
375     int temp_block_state;
376     int(*block_pointer)[4][4][4];
377     int temp_tetris_table[21][10];
378 
379     oldx = tempx = x;
380     oldy = tempy = y;
381     temp_block_state = block_state;
382 
383     switch (command)
384     {
385     case LEFT:
386         tempx--;
387         break;
388     case RIGHT:
389         tempx++;
390         break;
391     case DOWN:
392         tempy++;
393         break;
394     case ROTATE:
395         temp_block_state++;
396         temp_block_state %= 4;
397         break;
398     }
399 
400     switch (block_number)
401     {
402     case 0:
403         block_pointer = &i_block;
404         break;
405     case 1:
406         block_pointer = &t_block;
407         break;
408     case 2:
409         block_pointer = &s_block;
410         break;
411     case 3:
412         block_pointer = &z_block;
413         break;
414     case 4:
415         block_pointer = &l_block;
416         break;
417     case 5:
418         block_pointer = &j_block;
419         break;
420     case 6:
421         block_pointer = &o_block;
422         break;
423     }
424 
425     for (i = 0; i < 21; i++)
426     {
427         for (j = 0; j < 10; j++)
428         {
429             temp_tetris_table[i][j] = tetris_table[i][j];
430         }
431     }
432 
433     for (i = 0, oldy = y; i < 4; i++, oldy++)
434     {
435         for (j = 0, oldx = x; j < 4; j++, oldx++)
436         {
437             if (oldx > 0 && oldx < 9 && oldy < 20 && oldy > 0)
438             {
439                 if ((*block_pointer)[block_state][i][j] != 0)
440                     temp_tetris_table[oldy][oldx] = 0;
441             }
442         }
443     }
444 
445     for (i = 0; i < 4; i++)
446     {
447         for (j = 0; j < 4; j++)
448         {
```

```
449
450             if (temp_tetris_table[tempy + i][tempx + j] != 0 && (*block_pointer)[temp_block_state][i][j]
    != 0)
451                 return 1;
452         }
453     }
454
455     return 0;
456 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.11.3.3 display_tetris()

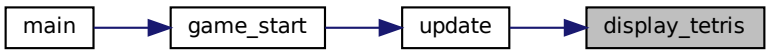
```
int display_tetris ( )
```

display_tetris.c 파일의 13 번째 라인에서 정의되었습니다.

```
14 {
15     int i, j;
16     int(*block_pointer)[4][4][4] = NULL;
17     int block_color; // 인트 블럭 색상 선언
18
19     switch (next_block_number)
20     {
21     case 0:
22         block_pointer = &i_block;
23         break;
24     case 1:
25         block_pointer = &t_block;
26         break;
27     case 2:
28         block_pointer = &s_block;
29         break;
30     case 3:
31         block_pointer = &z_block;
32         break;
33     case 4:
34         block_pointer = &l_block;
35         break;
36     case 5:
37         block_pointer = &j_block;
38         break;
39     case 6:
40         block_pointer = &o_block;
41         break;
42     }
43
44     // 터미널 화면을 지우고 다음 블록 출력
45     system("clear");
46     printf("Next Block\n");
47
48     //블록 출력 시작
49     for (i = 0; i < 4; i++)
50     {
51         for (j = 0; j < 4; j++)
52         {
53             // 블록이 있는 경우 해당 색상으로 출력합니다.
54             if ((*block_pointer)[0][i][j] == 1)
55             {
56                 printf(RED_COLOR "■ " RESET_COLOR);
57             }else if ((*block_pointer)[0][i][j] == 2){
58                 printf(GREEN_COLOR "■ " RESET_COLOR);
59             }else if ((*block_pointer)[0][i][j] == 3){
```

```
60                  printf(YELLOW_COLOR "■ " RESET_COLOR);
61              }else if ((*block_pointer)[0][i][j] == 4){
62                  printf(BLUE_COLOR "■ " RESET_COLOR);
63              }else if ((*block_pointer)[0][i][j] == 5){
64                  printf(MAGENTA_COLOR "■ " RESET_COLOR);
65              }else if ((*block_pointer)[0][i][j] == 6){
66                  printf(CYAN_COLOR "■ " RESET_COLOR);
67              }else if ((*block_pointer)[0][i][j] == 7){
68                  printf(WHITE_COLOR "■ " RESET_COLOR);
69              }else {
70                  printf("□ ");
71              }
72          }
73          printf("\n");
74      }
75      // 블록이 없는 경우 공백을 출력합니다.
76      printf("\n\n"); //두줄 공백 출력하는 것
77      printf("Tetris Table\n");
78      printf("\t┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬\n\t┌ ┬ ┬ ┬ ┬ ┬ ┐\n\t|  |  |  ∧  ∧  |  |    살려
        줘!!\n\t|  |  |  ( ≧ Д ≦ )  |  \n\t|  |  |  Φ  Φ  |  |\n\t└ ┴ ┴ ┴ ┴ ┴ ┘\n\t┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴\n");
79      for (i = 2; i < 21; i++)
80      {
81          printf("\t");
82          for (j = 0; j < 10; j++)
83          {
84              if (j == 0 || j == 9)
85              {
86                  printf("= ");
87              }
88              else if (i == 20 && (j > 0 && j < 9))
89              {
90                  printf("=");
91              }
92              else if (tetris_table[i][j] == 1){      //블럭포인터로 표현했었었는데, 테트리스테이블로 다시 함.
93                  printf(RED_COLOR "■ " RESET_COLOR);
94              }else if (tetris_table[i][j] == 2){
95                  printf(GREEN_COLOR "■ " RESET_COLOR);
96              }else if (tetris_table[i][j] == 3){
97                  printf(YELLOW_COLOR "■ " RESET_COLOR);
98              }else if (tetris_table[i][j] == 4){
99                  printf(BLUE_COLOR "■ " RESET_COLOR);
100             }else if (tetris_table[i][j] == 5){
101                 printf(MAGENTA_COLOR "■ " RESET_COLOR);
102             }else if (tetris_table[i][j] == 6){
103                 printf(CYAN_COLOR "■ " RESET_COLOR);
104             }else if (tetris_table[i][j] == 7){
105                 printf(WHITE_COLOR "■ " RESET_COLOR);
106             }else { // 블록이 없는 경우 공백
107                 printf("  ");
108             }
109         }
110         printf("\n");
111     }
112     printf("\t 신해철 - 그대에게\n\t0:35 ──────────── -3:48\n\t        II        \n");
113 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



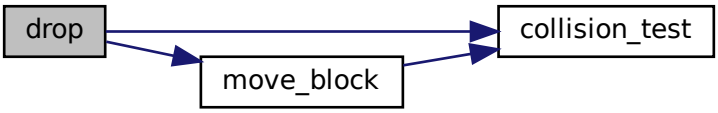
5.11.3.4 drop()

```
int drop (
            void  )
```

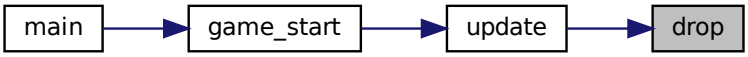
update.c 파일의 459 번째 라인에서 정의되었습니다.

```
460 {
461     while (!collision_test(DOWN))
462         move_block(DOWN);
463
464     return 0;
465 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.11.3.5 getch()

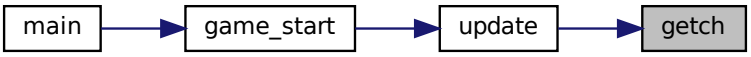
```
char getch ( )
```

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 }
```

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.11.3.6 move_block()

```
int move_block (
              int command )
```

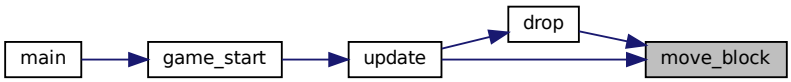
update.c 파일의 281 번째 라인에서 정의되었습니다.

```
282 {
283     int i, j;
284     int newx, newy;
285     int oldx, oldy;
286     int old_block_state;
287     int(*block_pointer)[4][4][4] = NULL;
288
289     newx = x;
290     newy = y;
291
292     old_block_state = block_state;
293
294     if (collision_test(command) == 0)
295     {
296         switch (command)
297         {
298         case LEFT:
299             newx--;
300             break;
301         case RIGHT:
302             newx++;
303             break;
304         case DOWN:
305             newy++;
306             break;
307         case ROTATE:
308             block_state++;
309             block_state %= 4;
310             break;
311         }
312     }
313     else
314     {
315         return 1;
316     }
317
318     switch (block_number)
319     {
320     case 0:
321         block_pointer = &i_block;
322         break;
323     case 1:
324         block_pointer = &t_block;
325         break;
326     case 2:
```

```
327         block_pointer = &s_block;
328         break;
329     case 3:
330         block_pointer = &z_block;
331         break;
332     case 4:
333         block_pointer = &l_block;
334         break;
335     case 5:
336         block_pointer = &j_block;
337         break;
338     case 6:
339         block_pointer = &o_block;
340         break;
341     }
342 
343         for (i = 0, oldy = y; i < 4; i++, oldy++)
344     {
345         for (j = 0, oldx = x; j < 4; j++, oldx++)
346         {
347             if (oldx > 0 && oldx < 9 && oldy < 20 && oldy > 0)
348                 if ((*block_pointer)[old_block_state][i][j] != 0)
349                     tetris_table[oldy][oldx] = 0;
350         }
351     }
352 
353     x = newx;
354     y = newy;
355 
356     for (i = 0, newy = y; i < 4; i++, newy++)
357     {
358         for (j = 0, newx = x; j < 4; j++, newx++)
359         {
360             if (newx > 0 && newx < 9 && newy < 20 && newy > 0)
361                 if ((*block_pointer)[block_state][i][j] != 0)
362                     tetris_table[newy][newx] = (*block_pointer)[block_state][i][j];
363         }
364     }
365 
366     return 0;
367 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:

이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.11.3.7 update()

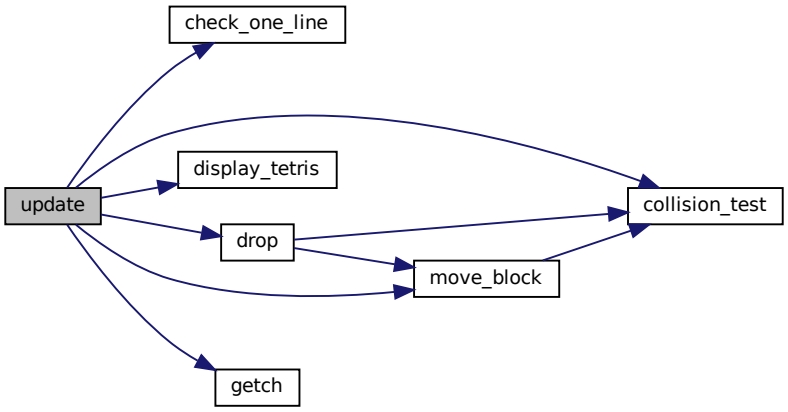
```
int update (
              int signum )
```

update.c 파일의 174 번째 라인에서 정의되었습니다.

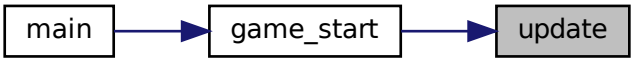
```
175 {
176     static int downcount = 0;
177     static int setcount = 0;
178     static long speedcount = 0;
179     static int countrange = 5;
180     static int firststart = 0;
181 
182     char ch;
183 
184     srand((unsigned)time(NULL));
185 
186     if (firststart == 0)
187     {
188         block_number = rand() % 7;
189         if (firststart == 0)
190             firststart++;
191     }
192 
193     display_tetris();
194     check_one_line();
195 
196     if (downcount == countrange - 1)
197     {
198         point += 1;
199         move_block(DOWN);
200     }
201 
202     if (speedcount == 499)
203     {
204         if (countrange != 1)
205             countrange--;
206     }
207 
208     downcount++;
209     downcount %= countrange;
210     speedcount++;
211     speedcount %= 500;
212 
213     if (x == 3 && y == 0)
214     {
215         if (collision_test(LEFT) || collision_test(RIGHT) || collision_test(DOWN) ||
      collision_test(ROTATE))
216         {
217             printf("\n Game End! \n");
218             downcount = 0;
219             setcount = 0;
220             speedcount = 0;
221             countrange = 5;
222             firststart = 0;
223             game = GAME_END;
224         }
225     }
226 
227     if (collision_test(DOWN))
228     {
229         if (setcount == 9)
230         {
231             block_number = next_block_number;
232             next_block_number = rand() % 7;
233             block_state = 0;
234             x = 3;
235             y = 0;
236         }
237         setcount++;
238         setcount %= 10;
239     }
240 
241     ch = getch();
242 
243     switch (ch)
244     {
245     case 74:
246     case 106:
247         move_block(LEFT);
248         break;
249     case 76:
250     case 108:
251         move_block(RIGHT);
```

```
252         break;
253     case 75:
254     case 107:
255         move_block(DOWN);
256         break;
257     case 73:
258     case 105:
259         move_block(ROTATE);
260         break;
261     case 65:
262     case 97:
263         drop();
264         break;
265     case 80:
266     case 112:
267         downcount = 0;
268         setcount = 0;
269         speedcount = 0;
270         countrange = 5;
271         firststart = 0;
272         game = GAME_END;
273         break;
274     default:
275         break;
276     }
277     return 0;
278 }
```

이 함수 내부에서 호출하는 함수들에 대한 그래프입니다.:



이 함수를 호출하는 함수들에 대한 그래프입니다.:



5.11.4 변수 문서화

5.11.4.1 block

```
int block[4][4][4]  [extern]
```

display_tetris.h 파일의 16 번째 라인에서 정의되었습니다.

5.11.4.2 block_number

```
int block_number
```

update.c 파일의 11 번째 라인에서 정의되었습니다.

5.11.4.3 block_state

```
int block_state = 0
```

update.c 파일의 7 번째 라인에서 정의되었습니다.

5.11.4.4 game

```
int game  [extern]
```

main.c 파일의 21 번째 라인에서 정의되었습니다.

5.11.4.5 i_block

```
int i_block[4][4][4]
```

초기값:

```
=
    {
        {{1, 1, 1, 1},
         {0, 0, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{0, 0, 0, 1},
         {0, 0, 0, 1},
         {0, 0, 0, 1},
         {0, 0, 0, 1}},
        {{0, 0, 0, 0},
         {0, 0, 0, 0},
         {1, 1, 1, 1},
         {0, 0, 0, 0}},
        {{1, 0, 0, 0},
         {1, 0, 0, 0},
         {1, 0, 0, 0},
         {1, 0, 0, 0}}
         }
```

update.c 파일의 17 번째 라인에서 정의되었습니다.

5.11.4.6 j_block

```
int j_block[4][4][4]
```

초기값:

```
=
    {
        {{0, 6, 0, 0},
         {0, 6, 0, 0},
         {6, 6, 0, 0},
         {0, 0, 0, 0}},
        {{6, 0, 0, 0},
         {6, 6, 6, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{6, 6, 0, 0},
         {6, 0, 0, 0},
         {6, 0, 0, 0},
         {0, 0, 0, 0}},
        {{6, 6, 6, 0},
         {0, 0, 6, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}

         }
```

update.c 파일의 122 번째 라인에서 정의되었습니다.

5.11.4.7 l_block

```
int l_block[4][4][4]
```

초기값:

```
=
    {
        {{5, 0, 0, 0},
         {5, 0, 0, 0},
         {5, 5, 0, 0},
         {0, 0, 0, 0}},
        {{5, 5, 5, 0},
         {5, 0, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{5, 5, 0, 0},
         {0, 5, 0, 0},
         {0, 5, 0, 0},
         {0, 0, 0, 0}},
        {{0, 0, 5, 0},
         {5, 5, 5, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}

         }
```

update.c 파일의 101 번째 라인에서 정의되었습니다.

5.11.4.8 next_block_number

```
int next_block_number
```

update.c 파일의 12 번째 라인에서 정의되었습니다.

5.11.4.9 o_block

```
int o_block[4][4][4]
```

초기값:

```
=
    {
        {{7, 7, 0, 0},
         {7, 7, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{7, 7, 0, 0},
         {7, 7, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{7, 7, 0, 0},
         {7, 7, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{7, 7, 0, 0},
         {7, 7, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}

         }
```

update.c 파일의 143 번째 라인에서 정의되었습니다.

5.11.4.10 point

```
int point  [extern]
```

main.c 파일의 25 번째 라인에서 정의되었습니다.

5.11.4.11 s_block

```
int s_block[4][4][4]
```

초기값:

```
=
    {
        {{3, 0, 0, 0},
         {3, 3, 0, 0},
         {0, 3, 0, 0},
         {0, 0, 0, 0}},
        {{0, 3, 3, 0},
         {3, 3, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{0, 3, 0, 0},
         {0, 3, 3, 0},
         {0, 0, 3, 0},
         {0, 0, 0, 0}},
        {{3, 3, 0, 0},
         {0, 3, 3, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}

         }
```

update.c 파일의 59 번째 라인에서 정의되었습니다.

5.11.4.12 t_block

```
int t_block[4][4][4]
```

초기값:
=
```
    {
        {{2, 0, 0, 0},
         {2, 2, 0, 0},
         {2, 0, 0, 0},
         {0, 0, 0, 0}},
        {{0, 2, 0, 0},
         {2, 2, 0, 0},
         {0, 2, 0, 0},
         {0, 0, 0, 0}},
        {{0, 0, 0, 0},
         {2, 2, 2, 0},
         {0, 2, 0, 0},
         {0, 0, 0, 0}},
        {{0, 2, 0, 0},
         {2, 2, 0, 0},
         {0, 2, 0, 0},
         {0, 0, 0, 0}}

         }
```

update.c 파일의 38 번째 라인에서 정의되었습니다.

5.11.4.13 tetris_table

```
int tetris_table[21][10]  [extern]
```

display_tetris.h 파일의 34 번째 라인에서 정의되었습니다.

5.11.4.14 x

```
int x  [extern]
```

main.c 파일의 23 번째 라인에서 정의되었습니다.

5.11.4.15 y

```
int y  [extern]
```

main.c 파일의 24 번째 라인에서 정의되었습니다.

5.11.4.16 z_block

```
int z_block[4][4][4]
```

초기값:

```
=
    {
        {{0, 4, 0, 0},
         {4, 4, 0, 0},
         {4, 0, 0, 0},
         {0, 0, 0, 0}},
        {{4, 4, 0, 0},
         {0, 4, 4, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{0, 0, 4, 0},
         {0, 4, 4, 0},
         {0, 4, 0, 0},
         {0, 0, 0, 0}},
        {{4, 4, 0, 0},
         {4, 0, 4, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}

         }
```

update.c 파일의 80 번째 라인에서 정의되었습니다.

Index