Датотека: game\_logic.c

|  |  |
| --- | --- |
| **Име функције** | startGame |
| **Прототип** | int startGame(SDL\_Texture\* user\_tank[TOTAL],SDL\_Texture\* bot\_tank1[TOTAL],SDL\_Texture\* bot\_tank2[TOTAL],SDL\_Texture\* map\_elements[TOTAL\_ELEMENTS]) |
| **Аргументи** | user\_tank [*in*]: низ текстура играчевог тенкића  bot\_tank1,bot\_tank2 [*in*]: низ текстура тенкића ботова  map\_elements [in]: низ текстура елемената мапе |
| **Повратна вредност** | 1 ако се излази из целе игре, 0 ако се само враћа у мени |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, SDL\_mixer.h, functions.h |
| **Пример коришћења** | Int done;  done=startGame(user\_tank, bot\_tank1, bot\_tank2, map\_elements);  if(done) exit(1); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | updateWall |
| **Прототип** | void updateWall(int cime, int\*\*map) |
| **Аргументи** | cime [*in*]: којим елементом попуњава зид око орла  map [*out*]: мапа |
| **Повратна вредност** | нема |
| **Захтева** | functions.h |
| **Пример коришћења** | case shovel: updateWall(STEEL, maps[lvl]); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | emptyStucts |
| **Прототип** | void emptyStructs(struct Tank tanks[MAX\_BOTS + 1], struct Bullet bullets[MAX\_BOTS + 1],int cordx,int cordy,struct Bullet\*bul2) |
| **Аргументи** | tanks [*out*]: низ свих тенкова  bullets [*out*]: низ свих метака  cordx[in]: почетна координата играча  cordy[in]: почетна координата играча  bul2[out]: други играчев метак уколико постоји |
| **Повратна вредност** | нема |
| **Захтева** | Functions.h |
| **Пример коришћења** | if (!(bots\_left) && !(botsAlive(tanks))) {  current\_level++;  emptyStructs(tanks, bullets, cordx, cordy,&bullet2);  bots\_left = MAX\_BOTS; } |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | createEnemie |
| **Прототип** | void createEnemie(struct Tank tanks[MAX\_BOTS + 1],int f\_left,int bot\_left) |
| **Аргументи** | F\_left[in]: да ли је слободан леви угао  Bot\_left[in]: број преоса  Tanks [out]: низ свих тенкова |
| **Повратна вредност** | нема |
| **Захтева** | functions.h |
| **Пример коришћења** | if ((bot\_create\_speed + (2000 \* botsAlive(tanks)) + ((MAX\_BOTS - bots\_left) % 10) \* 500<time\_check) && (bots\_left>0) && (areCornersClear(tanks, &f\_left, &f\_right))) {  createEnemie(tanks, f\_left, bots\_left);  bots\_left--;} |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | botsAlive |
| **Прототип** | int botsAlive(struct Tank tanks[MAX\_BOTS + 1]) |
| **Аргументи** | tanks [*in*]: низ свих тенкова |
| **Повратна вредност** | Број тренутно живих тенкића |
| **Захтева** | functions.h |
| **Пример коришћења** | if ((bot\_create\_speed + (2000 \* botsAlive(tanks)) + ((MAX\_BOTS - bots\_left) % 10) \* 500<time\_check) && (bots\_left>0) && (areCornersClear(tanks, &f\_left, &f\_right))) {  createEnemie(tanks, f\_left, bots\_left);  bots\_left--;} |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | areCornersClear |
| **Прототип** | int areCornersClear(struct Tank tanks[MAX\_BOTS + 1], int \*flag\_left, int\*flag\_right) |
| **Аргументи** | tanks [*in*]: низ свих тенкова  flag\_left [*out*]: да ли је леви угао чист  flag\_right [*out*]: да ли је десни угао чист |
| **Повратна вредност** | Да ли је један од углова слободан за рађање новог бота |
| **Захтева** | functions.h |
| **Пример коришћења** | if ((bot\_create\_speed + (2000 \* botsAlive(tanks)) + ((MAX\_BOTS - bots\_left) % 10) \* 500<time\_check) && (bots\_left>0) && (areCornersClear(tanks, &f\_left, &f\_right))) {  createEnemie(tanks, f\_left, bots\_left);  bots\_left--;} |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | newHighScore |
| **Прототип** | char\* newHighScore() |
| **Аргументи** | нема |
| **Повратна вредност** | Име које корисник укуца у новом прозору који се појави |
| **Захтева** | Functions.h |
| **Пример коришћења** | if (score > high\_scores[9].score) {  strcpy(high\_scores[9].username, newHighScore());  sort\_scores(high\_scores);  lock\_scores(high\_scores); |
| **Ограничења и проблеми** | Нема познатих проблема |

Датотека: AI.c

|  |  |
| --- | --- |
| **Име функције** | mapCopy |
| **Прототип** | int \*\*matCopy(int\*\* matrica, struct Tank tanks[MAX\_BOTS + 1]) |
| **Аргументи** | Matrica[in]: matrica koja rikazuje mapu  Tanks:niz koji sadrzi tenkice |
| **Повратна вредност** | Nova mapa na koju su ucrtani I tenkici |
| **Захтева** | Stdio.h |
| **Пример коришћења** | map=matCopy(matrica, tanks); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | gore, dole, levo, desno// ad isto za razlicite smerove |
| **Прототип** | int gore(struct Tank tenkic, int \*\*map, int target) |
| **Аргументи** | Map[in]: matrica koja rikazuje mapu  Tanks:niz koji sadrzi tenkice  Target:meta ovog tenkica |
| **Повратна вредност** | Vraca 1 ukoliko se u odredjenom pravcu nalazi meta, tenkic moze da puca na nju, 0 u suprotnom |
| **Захтева** | Stdio.h |
| **Пример коришћења** | return gore(tenkic, map, 8); |
| **Ограничења и проблеми** | Нема познатих проблема |
|  |  |

|  |  |
| --- | --- |
| **Име функције** | aheadTank,aheadPtica,aheadZid |
| **Прототип** | int aheadTank(struct Tank tenkic, int \*\*map) |
| **Аргументи** | Map[in]: matrica koja rikazuje mapu  Tanks:niz koji sadrzi tenkice |
| **Повратна вредност** | Vraca 1 ukoliko se pravo ispred tenkica nalazi protivnicki tenkic(ptica) |
| **Захтева** | Stdio.h |
| **Пример коришћења** | if ((aheadTank(tanks[i], map)) || (aheadPtica(tanks[i], map))) // ako su tenkic ili ptica ispred njega onda ce da puca  {  next\_moves = SHOOT;  return next\_moves;  } |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | bfs |
| **Прототип** | int bfs(int \*\*map, int tankx, int tanky, int task, int index, struct Tank tanks[MAX\_BOTS + 1], struct Bullet bullets[MAX\_BOTS + 1], int \*\*matrica)  { |
| **Аргументи** | Map[in]: matrica koja rikazuje mapu  Tanks[in]:niz koji sadrzi tenkice  Bullets[in]:niz koji sadrzi tenkice  Tankx[in]:x koordinata tenkica  Tanky[in]:y koordinata tenkica  Task[in]:zadatk tenkica  Inde[in]:redni broj tenkica u nizu tenkica |
| **Повратна вредност** | Odredjuje potez tenkica koji je optimalan, to znaci da je on na najkracem putu o mete. Najkraci put nalazi bfs. |
| **Захтева** | Stdio.h |
| **Пример коришћења** | next\_moves = bfs(map, tanks[i].x / UNIT, tanks[i].y / UNIT, task[i], i, tanks, bullets, matrica); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | dodge |
| **Прототип** | int dodge(struct Tank tenkic, struct Bullet metak, int \*\*map, int move) |
| **Аргументи** | Tenkic[in]:niz tenkica  Metak[in]: metak igraca  Map[in]:matrica koja sadrzi mapu  Move[in]:potez koji bi ovaj bot odigrao |
| **Повратна вредност** | Kaze da li bot ovim potezom nailazi na tenkic I tada vraca koji potez treba da odigra da bi metak izbegao. |
| **Захтева** | Stdio.h |
| **Пример коришћења** | izbegavanje = dodge(tanks[i], bullets[0], map, next\_moves) |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | easy |
| **Прототип** | int easy(struct Tank tanks[MAX\_BOTS + 1], struct Bullet bullets[MAX\_BOTS + 1], int\*\* map, int moduo, int i) |
| **Аргументи** | Tenks[in]:niz tenkica  bullets[in]: metak niz  Map[in]:matrica koja sadrzi mapu  Moduo[in]:koliko dugo traje igrica  I[in]:redni broj tenkica u nizu |
| **Повратна вредност** | Vraca koji potez treba lak bot da odigra |
| **Захтева** | Stdio.h |
| **Пример коришћења** | for (i = 1; i <= MAX\_BOTS; i++)  potez[i] = easy(tanks, bullets, map, moduo, i);  return potez; |
| **Ограничења и проблеми** | Нема познатих проблема |
|  |  |
| **Име функције** | medium |
| **Прототип** | int medium(struct Tank tanks[MAX\_BOTS + 1], struct Bullet bullets[MAX\_BOTS + 1], int\*\* matrica, int i) |
| **Аргументи** | Tenks[in]:niz tenkica  bullets[in]: metak niz  Matrica[in]:matrica koja sadrzi mapu  I[in]:redni broj tenkica u nizu |
| **Повратна вредност** | Vraca koji potez treba tezak bot da odigra |
| **Захтева** | Stdio.h |
| **Пример коришћења** | for (i = 1; i <= MAX\_BOTS; i++)  potez[i] = medium(tanks, bullets, map, i);  return potez; |
| **Ограничења и проблеми** | Нема познатих проблема |
| **Име функције** | premedium |
| **Прототип** | int\* premedium(struct Tank tanks[MAX\_BOTS + 1], struct Bullet bullets[MAX\_BOTS + 1], int\*\* matrica) |
| **Аргументи** | Tenks[in]:niz tenkica  bullets[in]: metak niz  Matrica[in]:matrica koja sadrzi mapu |
| **Повратна вредност** | Vraca koji potez treba srednji bot da odigra |
| **Захтева** | Stdio.h |
| **Пример коришћења** | if (tezina == 2)  {  return premedium(tanks, bullets, map, last\_turn);  } |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | hard |
| **Прототип** | int hard(struct Tank tanks[MAX\_BOTS + 1], struct Bullet bullets[MAX\_BOTS + 1], int\*\* matrica, int i) |
| **Аргументи** | Tenks[in]:niz tenkica  bullets[in]: metak niz  Matrica[in]:matrica koja sadrzi mapu  I[in]:redni broj tenkica |
| **Повратна вредност** | Vraca koji potez treba nemoguc bot da odigra |
| **Захтева** | Stdio.h |
| **Пример коришћења** | potez[i] = hard(tanks, bullets, map, i); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | Next\_moves |
| **Прототип** | int\* nextMoves(struct Tank tanks[MAX\_BOTS + 1], struct Bullet bullets[MAX\_BOTS + 1], int\*\* map) |
| **Аргументи** | Tenks[in]:niz tenkica  bullets[in]: metak niz  Map[in]:matrica koja sadrzi mapu |
| **Повратна вредност** | Vraca niz poteza koje treba botovi da odigraju |
| **Захтева** | Stdio.h |
| **Пример коришћења** | potez[i] = hard(tanks, bullets, map, i); |
| **Ограничења и проблеми** | Нема познатих проблема |

Датотека: menu\_background\_code.c

|  |  |
| --- | --- |
| **Име функције** | menu\_gui |
| **Прототип** | int menu\_gui() |
| **Аргументи** | нема |
| **Повратна вредност** | враћа изабрану опцију |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, SDL\_mixer.h, functions.h |
| **Пример коришћења** | choise = menu\_gui(); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | settingsScreen |
| **Прототип** | void settingsScreen(SDL\_Texture\* user\_tank[TOTAL], SDL\_Texture\* bot\_tank1[TOTAL], SDL\_Texture\* bot\_tank2[TOTAL], SDL\_Texture\* map\_elements[TOTAL\_ELEMENTS]) |
| **Аргументи** | user\_tank [*in*]: низ текстура играчевог тенкића  bot\_tank1,bot\_tank2 [*in*]: низ текстура тенкића ботова  map\_elements [in]: низ текстура елемената мапе |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, SDL\_mixer.h, functions.h |
| **Пример коришћења** | settingsScreen(user\_tank, bot\_tank1, bot\_tank2, map\_elements); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | highScoresScreen |
| **Прототип** | void highScoresScreen() |
| **Аргументи** | нема |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, SDL\_mixer.h, functions.h |
| **Пример коришћења** | highScoresScreen(); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | infoScreen |
| **Прототип** | void infoScreen() |
| **Аргументи** | нема |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, SDL\_mixer.h, functions.h |
| **Пример коришћења** | infoScreen(); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | createMaps |
| **Прототип** | void createMaps(SDL\_Texture\* map\_elements[TOTAL\_ELEMENTS]) |
| **Аргументи** | map\_elements [in]: низ текстура елемената мапе |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, SDL\_mixer.h, functions.h |
| **Пример коришћења** | createMaps(map\_elements[TOTAL\_ELEMENTS]); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | paintLevel |
| **Прототип** | void paintLevel(int \*\*map, SDL\_Texture\* map\_elements[TOTAL\_ELEMENTS], int x, int y) |
| **Аргументи** | map[*in*]: мапа која се црта  map\_elements [in]: низ текстура елемената мапе  x, y[*in*]: кординате за цртање нивоа у прозору |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, SDL\_mixer.h, functions.h |
| **Пример коришћења** | paintLevel(game\_maps[current\_map], map\_elements, x, y); |
| **Ограничења и проблеми** | Нема познатих проблема |

Датотека: game\_graphics.c

|  |  |
| --- | --- |
| **Име функције** | writeText |
| **Прототип** | void writeText(const char \*text, TTF\_Font \*g\_font, int x, int y, int height, int width, SDL\_Color front\_color, SDL\_Color back\_color) |
| **Аргументи** | text[*in*]: стринг који се исписује на екрану  g\_font[*in*]: фонт исписа  x, y, height, width[in]: позиције и димензије текста  front\_color, back\_color[in]: боја текста и боја позадине |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, functions.h |
| **Пример коришћења** | writeText(MENU, gFont, x, y, TEXT\_HEIGHT, strlen(MENU), TEXT\_COLOR, BACK\_COLOR); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | paintInfo |
| **Прототип** | void paintInfo() |
| **Аргументи** | нема |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, functions.h |
| **Пример коришћења** | paintInfo(); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | createHighScoreText |
| **Прототип** | char \*createHighScoreText(char \*user, int score) |
| **Аргументи** | user[*in*]: стринг корисничког имена  score[*in*]: резултат |
| **Повратна вредност** | стринг highScore који се састоји од размака, тачке, корисничког имена, размака и резултата |
| **Захтева** | string.h |
| **Пример коришћења** | high\_scores = createHighScoreText(high\_scores[i].username, high\_scores[i].score); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | paintHighScores |
| **Прототип** | void paintHighScores() |
| **Аргументи** | нема |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, functions.h |
| **Пример коришћења** | paintHighScores(); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | paintMap |
| **Прототип** | void paintMap(int\*\*\* all\_maps, int current\_level, int bots\_left, int score, int lives, struct PowerUp powerup, struct Tank tanks[MAX\_BOTS + 1], struct Bullet bullets[MAX\_BOTS + 1], SDL\_Texture\* player[TOTAL], SDL\_Texture\* bot1[TOTAL], SDL\_Texture\* bot\_spec[TOTAL], SDL\_Texture\* map\_elements[TOTAL\_ELEMENTS]) { |
| **Аргументи** | all\_maps[*in*]: стринг који се исписује на екрану  current\_level, bots\_left, score, lives[*in*]: тренутни ниво, колико је остало ботова, резултат и број живота  powerup[*in*]: тип и координате појачања  tanks[*in*]: низ тенкића са координатама смером и снагом  bullets[*in*]: низ метака са координатама и смером  player, bot1, bot\_spec, map\_elements[*in*]: текстуре за играча, ботове и елементе мапе |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, functions.h |
| **Пример коришћења** | paintMap(all\_maps, current\_level, bots\_left + botsAlive(tanks), score, player\_lives, power\_up, tanks, bullets, user\_tank, bot\_tank1, bot\_tank2, map\_elements); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | paintSettings |
| **Прототип** | void paintSettings(int choice) |
| **Аргументи** | choise[*in*]: одређује на којој позицији се црта стрелица |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, functions.h |
| **Пример коришћења** | void paintSettings(choice); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | paintMenu |
| **Прототип** | void paintMenu(int choice) |
| **Аргументи** | choise[*in*]: одређује на којој позицији се црта стрелица |
| **Повратна вредност** | нема |
| **Захтева** | SDL.h, SDL\_image.h, SDL\_ttf.h, functions.h |
| **Пример коришћења** | void paintMenu(choice); |
| **Ограничења и проблеми** | Нема познатих проблема |

Датотека: scores.c

|  |  |
| --- | --- |
| **Име функције** | encrypt |
| **Прототип** | void encrypt(unsigned long int \* v, unsigned long int\* k) |
| **Аргументи** | unsigned long int \* v низ од 2 елемента  unsigned long int\* k низ од 4 елемента |
| **Повратна вредност** | нема |
| **Захтева** | functions.h |
| **Пример коришћења** | Енкриптује 2 елемента низа  encrypt(&S[18],k); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | decrypt |
| **Прототип** | void decrypt(unsigned long int \* v, unsigned long int\* k) |
| **Аргументи** | unsigned long int \* v низ од 2 елемента  unsigned long int\* k низ од 4 елемента |
| **Повратна вредност** | нема |
| **Захтева** | functions.h |
| **Пример коришћења** | Декриптује 2 елемента низа  decrypt(&S[18],k); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | starting\_scores |
| **Прототип** | SCORES \*starting\_scores() |
| **Аргументи** | нема |
| **Повратна вредност** | SCORES skor[] |
| **Захтева** | Functions.h |
| **Пример коришћења** | Узима потребне из датотеке.  starting\_scores(); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | lock\_scores |
| **Прототип** | void lock\_scores(SCORES \*skor) |
| **Аргументи** | SCORES skor[] збирка скорова које треба сачувати |
| **Повратна вредност** | нема |
| **Захтева** | functions.h |
| **Пример коришћења** | lock\_scores(scores); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | sort\_scores |
| **Прототип** | void sort\_scores(SCORES \*skor) |
| **Аргументи** | SCORES skor[] збирка скорова |
| **Повратна вредност** | Нема |
| **Захтева** | functions.h |
| **Пример коришћења** | sort\_scores(scores); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | restore |
| **Прототип** | void restore() |
| **Аргументи** | Нема |
| **Повратна вредност** | Нема |
| **Захтева** | functions.h |
| **Пример коришћења** | restore(); |
| **Ограничења и проблеми** | Нема познатих проблема |

Датотека: game\_maps.c

|  |  |
| --- | --- |
| **Име функције** | CleanSpots |
| **Прототип** | void cleanSpots(int \*\*\*a) |
| **Аргументи** | Int \*\*\*a збирка нивоа |
| **Повратна вредност** | нема |
| **Захтева** | functions.h |
| **Пример коришћења** | Чисти одређена поља да би се омогућило кретање тенкића  CleanSpots(a); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | INSERT\_O |
| **Прототип** | void INSERT\_O(int \*\*mat, int m, int n, int x) |
| **Аргументи** | int \*\*mat ниво  int m координата  int n координата  int x врста поља |
| **Повратна вредност** | нема |
| **Захтева** | functions.h |
| **Пример коришћења** | Попуњава блок 4x4 одређеном врстом поља  INSERT\_O(a[4], 4, 4, BRICK); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | INSERT\_OО |
| **Прототип** | void INSERT\_OО(int \*\*mat, int m, int n, int x) |
| **Аргументи** | int \*\*mat ниво  int m координата  int n координата  int x врста поља |
| **Повратна вредност** | нема |
| **Захтева** | Functions.h |
| **Пример коришћења** | INSERT\_OO(a[4], 4, 4, BRICK);  Попуњава блок 2x2 одређеном врстом поља |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | random\_mapa |
| **Прототип** | void random\_mapa(int \*\*\*a, int lvl) |
| **Аргументи** | int \*\*\*a збирка нивоа  int lvl конкретан ниво |
| **Повратна вредност** | нема |
| **Захтева** | functions.h |
| **Пример коришћења** | If (current\_level>MAX\_LEVELS) random\_mapa(a,current\_level); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | create\_map |
| **Прототип** | int \*\*\* create\_map() |
| **Аргументи** | Нема |
| **Повратна вредност** | int \*\*\*a збирка нивоа, празне мапе |
| **Захтева** | functions.h |
| **Пример коришћења** | позива се на почетку уколико су потребне празне мапе  create\_map(); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | doteraj\_mape |
| **Прототип** | void doteraj\_mape(int \*\*\*a, int lvl) |
| **Аргументи** | int \*\*\*a збирка нивоа  int lvl конкретан ниво |
| **Повратна вредност** | Нема |
| **Захтева** | functions.h |
| **Пример коришћења** | random\_mapa {  …  doteraj\_mape(a,lvl);  } |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | upisi |
| **Прототип** | void upisi(int \*\*\*a) |
| **Аргументи** | int \*\*\*a |
| **Повратна вредност** |  |
| **Захтева** | Functions.h |
| **Пример коришћења** | Фја за чување мапа у датотеку  upisi(a); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | ispisi |
| **Прототип** | void ispisi (int \*\*\*a) |
| **Аргументи** | int \*\*\*a |
| **Повратна вредност** |  |
| **Захтева** | Functions.h |
| **Пример коришћења** | Фја за узимање података о мапама из датотеке  ispisi(a); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | getMaps2 |
| **Прототип** | int \*\*\*getMaps2(int\*\*\*a) |
| **Аргументи** | int \*\*\*a |
| **Повратна вредност** | int \*\*\*a |
| **Захтева** | Functions.h |
| **Пример коришћења** | Попуни празне мапе или направи рандом мапу.  a=getMaps2(a); |
| **Ограничења и проблеми** | Нема познатих проблема |

|  |  |
| --- | --- |
| **Име функције** | dealok |
| **Прототип** | void dealok(int \*\*\*a) |
| **Аргументи** | int \*\*\*a |
| **Повратна вредност** | Нема |
| **Захтева** | Functions.h |
| **Пример коришћења** | dealok(a); |
| **Ограничења и проблеми** | Нема познатих проблема |