Rubik Solver Group 4

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 cli_handler Namespace Reference

Classes

- class SolverCLI

  Klasa zarządzająca komunikacją CLI i integracją.

Variables

- BASE_DIR = Path(__file__).resolve().parent.parent
- string DATA_DIR = BASE_DIR / "data"
- string INPUT_FILE = DATA_DIR / "cube_state.txt"
- string OUTPUT_FILE = DATA_DIR / "solution_steps.txt"
- list FACES_ORDER = ['U', 'D', 'F', 'B', 'L', 'R']
- dictionary VALID_COLORS = {'W', 'Y', 'G', 'B', 'O', 'R'}
- cli = SolverCLI()

4.1.1 Variable Documentation

4.1.1.1 BASE_DIR

```
cli_handler.BASE_DIR = Path(__file__).resolve().parent.parent
```

4.1.1.2 cli

```
cli_handler.cli = SolverCLI()
```

4.1.1.3 DATA_DIR

```
string cli_handler.DATA_DIR = BASE_DIR / "data"
```

4.1.1.4 FACES_ORDER

```
list cli_handler.FACES_ORDER = ['U', 'D', 'F', 'B', 'L', 'R']
```

4.1.1.5 INPUT_FILE

```
string cli_handler.INPUT_FILE = DATA_DIR / "cube_state.txt"
```

4.1.1.6 OUTPUT_FILE

```
string cli_handler.OUTPUT_FILE = DATA_DIR / "solution_steps.txt"
```

4.1.1.7 VALID_COLORS

```
dictionary cli_handler.VALID_COLORS = {'W', 'Y', 'G', 'B', 'O', 'R'}
```

4.2 input_gui Namespace Reference

Classes

- class InputGUI

Variables

- dictionary COLORS
- dictionary HEX_TO_NAME = {v: k for k, v in COLORS.items()}
- dictionary POLISH_NAME
- root = tk.Tk()
- cube_editor = InputGUI(root)

4.2.1 Variable Documentation

4.2.1.1 COLORS

```
dictionary input_gui.COLORS
```

Initial value:

```
1 =  {
2     'white':  '#FFFFFF',
3     'yellow': '#FFFF00',
4     'green':  '#00FF00',
5     'blue':   '#0000FF',
6     'red':    '#FF0000',
7     'orange': '#FFA500',
8     'grey':   '#D3D3D3'
9 }
```

4.2.1.2 cube_editor

```
input_gui.cube_editor = InputGUI(root)
```

4.2.1.3 HEX_TO_NAME

```
dictionary input_gui.HEX_TO_NAME = {v:  k for k, v in COLORS.items()}
```

4.2.1.4 POLISH_NAME

```
dictionary input_gui.POLISH_NAME
```

Initial value:

```
1 =  {
2     'white':  'biały',
3     'yellow': 'żółty',
4     'green':  'zielony',
5     'blue':   'niebieski',
6     'red':    'czerwony',
7     'orange': 'pomarańczowy',
8     'grey':   'szary'
9 }
```

4.2.1.5 root

```
input_gui.root = tk.Tk()
```

4.3 main Namespace Reference

Functions

- def run_cpp_solver ()

  Uruchamia solver napisany w C++.
- def run_gui ()

  Uruchamia wizualizację rozwiązania.
- def main ()

  Główna funkcja programu.

Variables

- BASE_DIR = Path(__file__).resolve().parent
- string CPP_EXEC = BASE_DIR / "src_cpp" / "main"
- string GUI_SCRIPT = BASE_DIR / "src_py" / "vis_gui.py"

4.3.1 Function Documentation

4.3.1.1 main()

```
def main.main ( )
```

Główna funkcja programu.

Parsuje argumenty wiersza poleceń, steruje przepływem danych między modułami inputu, solvera i wizualizacji.

4.3.1.2 run_cpp_solver()

```
def main.run_cpp_solver ( )
```

Uruchamia solver napisany w C++.

Wywołuje plik wykonywalny przekazując stan kostki przez stdin i zapisując wynik do stdout. Ustawia katalog roboczy na src_cpp, aby solver poprawnie znalazł heurystyki w ../data.

Returns

True jeśli solver zakończył się sukcesem, False w przeciwnym razie.

```
Runs the C++ solver executable.
```

4.3.1.3 run_gui()

```
def main.run_gui ( )
```

Uruchamia wizualizację rozwiązania.

Kopiuje wynik solvera do pliku test.txt (wymagane przez vis_gui.py) i uruchamia skrypt wizualizacji.

```
Runs the visualization GUI.
```

4.3.2 Variable Documentation

4.3.2.1 BASE_DIR

```
main.BASE_DIR = Path(__file__).resolve().parent
```

4.3.2.2 CPP_EXEC

```
string main.CPP_EXEC = BASE_DIR / "src_cpp" / "main"
```

4.3.2.3 GUI_SCRIPT

```
string main.GUI_SCRIPT = BASE_DIR / "src_py" / "vis_gui.py"
```

4.4 vis_gui Namespace Reference

Functions

- def load_data (filename)

  wczytuje kolejne stany kostki z pliku tekstowego
- def build_cube (cube_state)

  Buduje trójwymiarowy model kostki Rubika.
- def draw_cubie (ax, x, y, z, colours)

  Rysuje pojedynczy element kostki Rubika w przestrzeni 3D.
- def update (frame)

  Aktualizuje pojedynczą klatkę animacji.

Variables

- dictionary colour_map
- def moves = load_data("test.txt")
- fig = plt.figure(figsize=(10,10))
- list axes
- ani = FuncAnimation(fig, update, frames=len(moves), interval=1000)
- writer
- fps

4.4.1 Function Documentation

4.4.1.1 build_cube()

```
def vis_gui.build_cube (
              cube_state )
```

Buduje trójwymiarowy model kostki Rubika.

Funkcja tworzy listę 27 małych sześcianów na podstawie stanu kostki, określając ich współrzędne oraz kolory widocznych ścian.

Parameters

Słownik	opisujący aktualny stan kostki

Returns

Lista krotek (x, y, z, colours), gdzie colours oznacza słownik kolorów ścian

4.4.1.2 draw_cubie()

```
def vis_gui.draw_cubie (
              ax,
              x,
              y,
              z,
              colours )
```

Rysuje pojedynczy element kostki Rubika w przestrzeni 3D.

Funkcja rysuje ściany pojedynczego sześcianu na podanej osi 3D. param ax Oś matplotlib 3D, na której rysowany jest sześcian. param x Współrzędna x sześcianu param y Współrzędna y sześcianu param z Współrzędna z sześcianu param colours Słownik kolorów widocznych ścian sześcianu.

Returns

Lista obiektów Poly3DCollection odpowiadających narysowanym ścianom.

4.4.1.3 load_data()

```
def vis_gui.load_data (
              filename )
```

wczytuje kolejne stany kostki z pliku tekstowego

Funkcja odczytuje dane z pliku i zamienia je na liste słowników. Każdy stan jest zapisywany jako słownik zawierający sześć ścian (U, D, F, B, L, R), z których każda opisana jest tablicą 3x3 kolorów

Parameters

filename	Ścieżka do pliku tekstowego z danymi wejściowymi.

Returns

Lista słowników, gdzie każdy element opisuje jeden pełny stan kostki.

4.4.1.4 update()

```
def vis_gui.update (
              frame )
```

Aktualizuje pojedynczą klatkę animacji.

Dla podanej klatki czyści osie i rysuje aktualny stan kostki z czterech różnych perspektyw param Numer aktualnej klatki return Pusta lista, wymagana przez mechanizm FuncAnimation.

4.4.2 Variable Documentation

4.4.2.1 ani

```
vis_gui.ani = FuncAnimation(fig, update, frames=len(moves), interval=1000)
```

4.4.2.2 axes

```
list vis_gui.axes
```

Initial value:

```
1 = [
2     fig.add_subplot(2,2,1, projection='3d'),
3     fig.add_subplot(2,2,2, projection='3d'),
4     fig.add_subplot(2,2,3, projection='3d'),
5     fig.add_subplot(2,2,4, projection='3d')
6 ]
```

4.4.2.3 colour_map

dictionary vis_gui.colour_map

Initial value:

```
1 = {
2     'W': 'white',
3     'Y': 'yellow',
4     'G': 'green',
5     'B': 'blue',
6     'O': 'orange',
7     'R': 'red'
8 }
```

4.4.2.4 fig

vis_gui.fig = plt.figure(figsize=(10,10))

4.4.2.5 fps

vis_gui.fps

4.4.2.6 moves

def vis_gui.moves = load_data("test.txt")

4.4.2.7 writer

vis_gui.writer

Chapter 5

Class Documentation

5.1 cph Struct Reference

`#include <cph.h>`

Public Member Functions

- cph ()
- int get_cph (cube &state)

Public Attributes

- int _cph [40320]

5.1.1 Constructor & Destructor Documentation

5.1.1.1 cph()

```
cph::cph ( )
```

5.1.2 Member Function Documentation

5.1.2.1 get_cph()

```
int cph::get_cph (
            cube & state )
```

5.1.3 Member Data Documentation

5.1.3.1 _cph

```
int cph::_cph[40320]
```

The documentation for this struct was generated from the following files:

- src_cpp/include/cph.h
- src_cpp/cph.cpp

5.2 cube Class Reference

```
#include <cube.h>
```

Public Member Functions

- cube ()
- void R ()
- void L ()
- void U ()
- void D ()
- void B ()
- void F ()
- bool read ()
- void print ()
- void move (string id)

Public Attributes

- array< uint8_t, 8 > cp
- array< uint8_t, 8 > co
- array< uint8_t, 12 > ep
- array< uint8_t, 12 > eo

5.2.1 Constructor & Destructor Documentation

5.2.1.1 cube()

```
cube::cube ( )
```

5.2.2 Member Function Documentation

5.2.2.1 B()

```
void cube::B ( )
```

5.2.2.2 D()

```
void cube::D ( )
```

5.2.2.3 F()

```
void cube::F ( )
```

5.2.2.4 L()

```
void cube::L ( )
```

5.2.2.5 move()

```
void cube::move (
            string id )
```

5.2.2.6 print()

```
void cube::print ( )
```

5.2.2.7 R()

```
void cube::R ( )
```

5.2.2.8 read()

```
bool cube::read ( )
```

5.2.2.9 U()

```
void cube::U ( )
```

5.2.3 Member Data Documentation

5.2.3.1 co

```
array<uint8_t, 8> cube::co
```

5.2.3.2 cp

```
array<uint8_t, 8> cube::cp
```

5.2.3.3 eo

```
array<uint8_t, 12> cube::eo
```

5.2.3.4 ep

```
array<uint8_t, 12> cube::ep
```

The documentation for this class was generated from the following files:

- src_cpp/include/cube.h
- src_cpp/cube.cpp

5.3 eoh Struct Reference

```
#include <eoh.h>
```

Public Member Functions

- eoh ()
- int get_eoh (cube &state)

Public Attributes

- int _eoh [4096]

5.3.1 Constructor & Destructor Documentation

5.3.1.1 eoh()

```
eoh::eoh ( )
```

5.3.2 Member Function Documentation

5.3.2.1 get_eoh()

```
int eoh::get_eoh (
            cube & state )
```

5.3.3 Member Data Documentation

5.3.3.1 _eoh

```
int eoh::_eoh[4096]
```

The documentation for this struct was generated from the following files:

- src_cpp/include/eoh.h
- src_cpp/eoh.cpp

5.4 eph Struct Reference

```
#include <eph.h>
```

Public Member Functions

- eph ()
- int get_eph (cube &state)

Public Attributes

- int _eph [479001600]

5.4.1 Constructor & Destructor Documentation

5.4.1.1 eph()

```
eph::eph ( )
```

5.4.2 Member Function Documentation

5.4.2.1 get_eph()

```
int eph::get_eph (
            cube & state )
```

5.4.3 Member Data Documentation

5.4.3.1 _eph

```
int eph::_eph[479001600]
```

The documentation for this struct was generated from the following files:

- src_cpp/include/eph.h
- src_cpp/eph.cpp

5.5 input_gui.InputGUI Class Reference

Public Member Functions

- def __init__ (self, root)

  Główna klasa zarządzająca interfejsem graficznym.
- def create_palette (self)

  Tworzy górny pasek z paletą kolorów i przyciskiem instrukcji.
- def show_instructions (self)

  Wyświetla okno dialogowe z instrukcją obsługi programu.
- def set_brush (self, color)

  Ustawia aktualnie używany kolor "pędzla".
- def create_grid (self)

  Konfiguruje układ siatki i rozmieszcza ścianki kostki.
- def create_face (self, face_name, offset_x, offset_y)

  Tworzy siatkę 3x3 przycisków dla pojedynczej ścianki.
- def paint_tile (self, button_widget)

  Zmienia kolor tła kliknięgo przycisku.
- def verify_and_save (self)

  Weryfikuje poprawność stanu kostki i inicjuje zapis.
- def save_to_file (self, data)

  Zapisuje przetworzone dane do pliku cube_input.txt.

Public Attributes

- root
- current_color
- tiles
- grid_frame

5.5.1 Constructor & Destructor Documentation

5.5.1.1 __init__()

```
def input_gui.InputGUI.__init__ (
              self,
              root )
```

Główna klasa zarządzająca interfejsem graficznym.

Konstruktor klasy InputGUI.

Inicjalizuje główne okno, zmienne stanu i buduje układ interfejsu.

Parameters

root	Obiekt głównego okna tkinter (tk.Tk).

5.5.2 Member Function Documentation

5.5.2.1 create_face()

```
def input_gui.InputGUI.create_face (
              self,
              face_name,
              offset_x,
              offset_y )
```

Tworzy siatkę 3x3 przycisków dla pojedynczej ścianki.

Obsługuje logikę blokowania środków ścian (center tiles).

Parameters

face_name	Symbol ścianki (np. 'U', 'F').
offset_x	Pozycja kolumny w głównej siatce.
offset_y	Pozycja wiersza w głównej siatce.

5.5.2.2 create_grid()

```
def input_gui.InputGUI.create_grid (
              self )
```

Konfiguruje układ siatki i rozmieszcza ścianki kostki.

Definiuje mapowanie nazw ścianek (U, L, F, R, B, D) na pozycje (x, y).

5.5.2.3 create_palette()

```
def input_gui.InputGUI.create_palette (
              self )
```

Tworzy górny pasek z paletą kolorów i przyciskiem instrukcji.

Generuje przyciski dla każdego koloru zdefiniowanego w COLORS (oprócz szarego).

5.5.2.4 paint_tile()

```
def input_gui.InputGUI.paint_tile (
              self,
              button_widget )
```

Zmienia kolor tła kliknięty przycisku.

Parameters

button_widget	Obiekt przycisku, który został kliknięty.

5.5.2.5 save_to_file()

```
def input_gui.InputGUI.save_to_file (
              self,
              data )
```

Zapisuje przetworzone dane do pliku cube_input.txt.

Dokonuje transformacji danych (mapowanie nazw na litery) oraz obraca ściankę 'U' o 180 stopni zgodnie z wymaganiami algorytmu.

Parameters

data	Słownik zawierający stan kostki { (face, r, c): color_name }.

5.5.2.6 set_brush()

```
def input_gui.InputGUI.set_brush (
              self,
              color )
```

Ustawia aktualnie używany kolor "pędzla".

Parameters

color	Nazwa koloru (klucz ze słownika COLORS).

5.5.2.7 show_instructions()

```
def input_gui.InputGUI.show_instructions (
              self )
```

Wyświetla okno dialogowe z instrukcją obsługi programu.

5.5.2.8 verify_and_save()

```
def input_gui.InputGUI.verify_and_save (
              self )
```

Weryfikuje poprawność stanu kostki i inicjuje zapis.

Sprawdza czy nie ma szarych pól oraz czy każdy kolor występuje dokładnie 9 razy. Jeśli weryfikacja przebiegnie pomyślnie, wywołuje save_to_file() i zamyka program.

5.5.3 Member Data Documentation

5.5.3.1 current_color

```
input_gui.InputGUI.current_color
```

5.5.3.2 grid_frame

```
input_gui.InputGUI.grid_frame
```

5.5.3.3 root

```
input_gui.InputGUI.root
```

5.5.3.4 tiles

```
input_gui.InputGUI.tiles
```

The documentation for this class was generated from the following file:

- src_py/input_gui.py

5.6 cli_handler.SolverCLI Class Reference

Klasa zarządzająca komunikacją CLI i integracją.

Public Member Functions

- def __init__ (self, timeout_sec=5.0)

  Inicjalizacja z domyślnym timeoutem.
- def get_user_input (self)

  Interaktywnie pobiera stan kostki od użytkownika.
- def save_to_file (self, cube_string)

  Zapisuje sformatowany stan kostki do pliku.
- def run (self)

  Główna pętla aplikacji.

Public Attributes

- timeout

5.6.1 Detailed Description

Klasa zarządzająca komunikacją CLI i integracją.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 __init__()

```
def cli_handler.SolverCLI.__init__ (
              self,
              timeout_sec = 5.0 )
```

Inicjalizacja z domyślnym timeoutem.

5.6.3 Member Function Documentation

5.6.3.1 get_user_input()

```
def cli_handler.SolverCLI.get_user_input (
              self )
```

Interaktywnie pobiera stan kostki od użytkownika.

Pyta oddzielnie o każdą z 6 ścianek, walidując długość i znaki.

Returns

Sformatowany ciąg gotowy do zapisu do pliku lub None w przypadku przerwania.

5.6.3.2 run()

```
def cli_handler.SolverCLI.run (
              self )
```

Główna pętla aplikacji.

5.6.3.3 save_to_file()

```
def cli_handler.SolverCLI.save_to_file (
              self,
              cube_string )
```

Zapisuje sformatowany stan kostki do pliku.

Parameters

cube_string	Gotowy ciąg danych.

Returns

True jeśli zapis się powiódł.

5.6.4 Member Data Documentation

5.6.4.1 timeout

```
cli_handler.SolverCLI.timeout
```

The documentation for this class was generated from the following file:

- src_py/cli_handler.py

Chapter 6

File Documentation

6.1 main.py File Reference

Namespaces

- main

Functions

- def main.run_cpp_solver ()

  Uruchamia solver napisany w C++.
- def main.run_gui ()

  Uruchamia wizualizację rozwiązania.
- def main.main ()

  Główna funkcja programu.

Variables

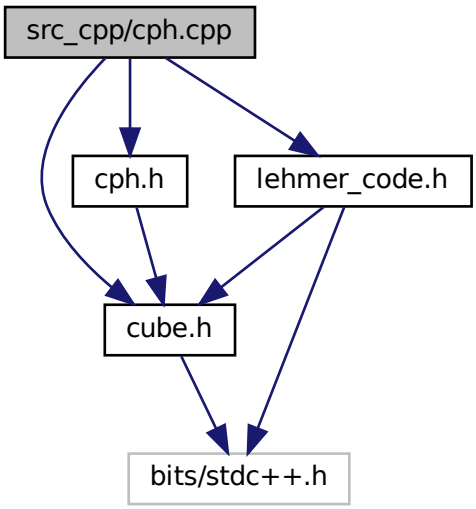
- main.BASE_DIR = Path(__file__).resolve().parent
- string main.CPP_EXEC = BASE_DIR / "src_cpp" / "main"
- string main.GUI_SCRIPT = BASE_DIR / "src_py" / "vis_gui.py"

6.2 src_cpp/cph.cpp File Reference

```
#include "cube.h"
#include "cph.h"
```

```
#include "lehmer_code.h"
```

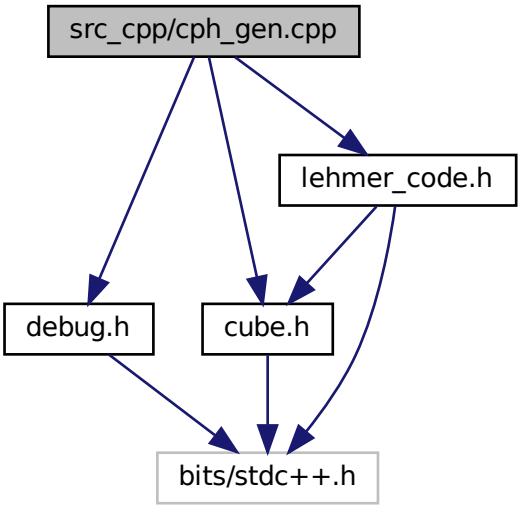
Include dependency graph for cph.cpp:



6.3 src_cpp/cph_gen.cpp File Reference

```
#include "debug.h"
#include "cube.h"
#include "lehmer_code.h"
```

Include dependency graph for cph_gen.cpp:



Functions

- int main ()

6.3.1 Function Documentation

6.3.1.1 main()

```
int main ( )
```

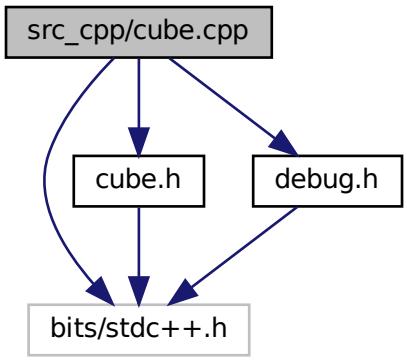
6.4 src_cpp/cube.cpp File Reference

Implementacja klasy cube.

```
#include <bits/stdc++.h>
#include "cube.h"
```

```
#include "debug.h"
```

Include dependency graph for cube.cpp:



6.4.1 Detailed Description

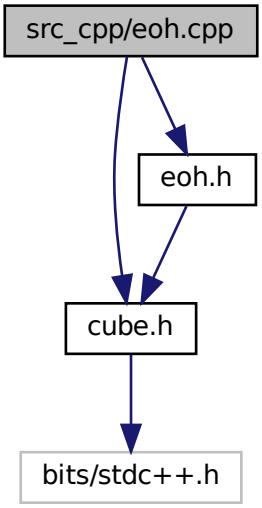
Implementacja klasy cube.

Zawiera metody do manipulacji stanem kostki (obroty).

6.5 src_cpp/eoh.cpp File Reference

```
#include "cube.h"
#include "eoh.h"
```

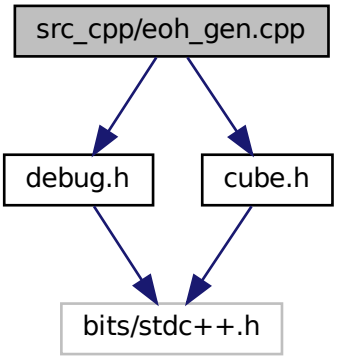
Include dependency graph for eoh.cpp:



6.6 src_cpp/eoh_gen.cpp File Reference

```
#include "debug.h"
#include "cube.h"
```

Include dependency graph for eoh_gen.cpp:



Functions

- int main ()

6.6.1 Function Documentation

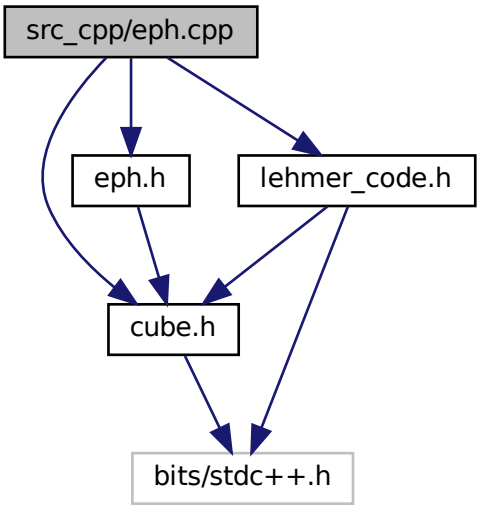
6.6.1.1 main()

```
int main ( )
```

6.7 src_cpp/eph.cpp File Reference

```
#include "cube.h"
#include "eph.h"
#include "lehmer_code.h"
```

Include dependency graph for eph.cpp:

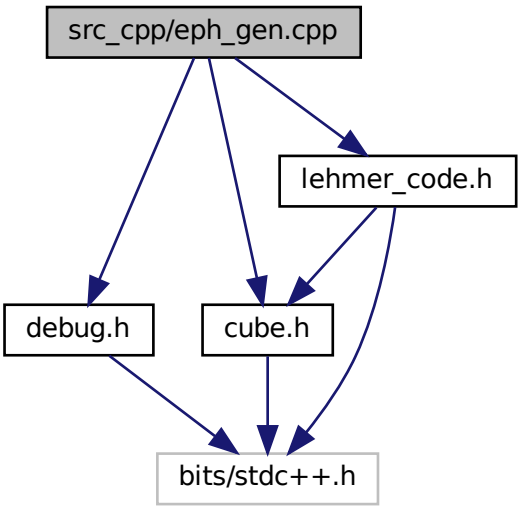


6.8 src_cpp/eph_gen.cpp File Reference

```
#include "debug.h"
#include "cube.h"
```

```
#include "lehmer_code.h"
```

Include dependency graph for eph_gen.cpp:



Functions

- int main ()

6.8.1 Function Documentation

6.8.1.1 main()

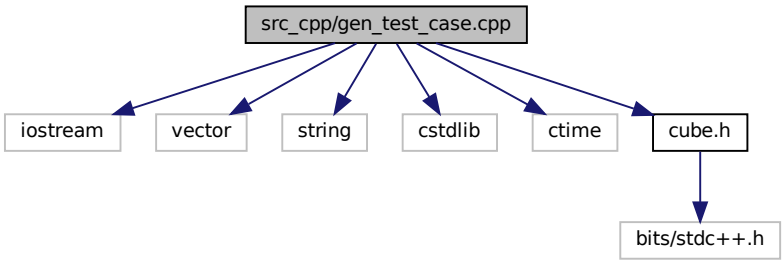
```
int main ( )
```

6.9 src_cpp/gen_test_case.cpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <cstdlib>
#include <ctime>
```

```
#include "cube.h"
```

Include dependency graph for gen_test_case.cpp:



Functions

- int main ()

6.9.1 Function Documentation

6.9.1.1 main()

```
int main ( )
```

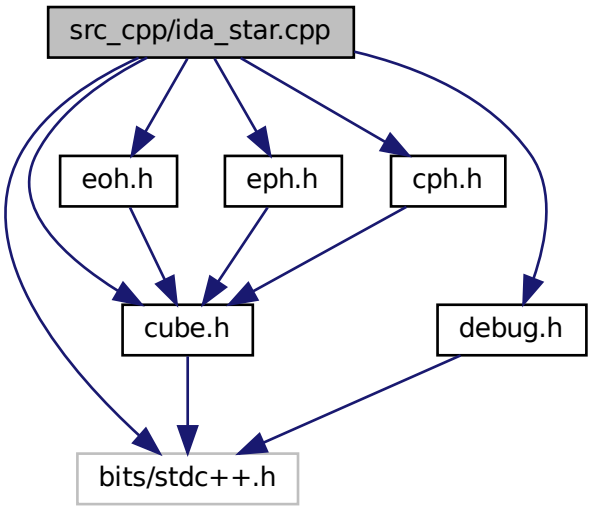
6.10 src_cpp/ida_star.cpp File Reference

Implementacja algorytmu IDA∗.

```
#include <bits/stdc++.h>
#include "cube.h"
#include "debug.h"
#include "eoh.h"
#include "eph.h"
```

```
#include "cph.h"
```

Include dependency graph for ida_star.cpp:



Macros

- #define inf 1000000000

  Stała reprezentująca „nieskończoność” używaną w IDA∗.

Typedefs

- typedef long long LL

Functions

- bool is_goal_phase2 (cube &c)

  Sprawdza, czy kostka jest rozwiązana w fazie 2.
- bool is_goal_phase1 (cube &c)

  Sprawdza, czy kostka spełnia warunki końcowe fazy 1.
- int getheuristic (cube &node, int stage)

  Oblicza wartość heurystyki dla danego stanu kostki.
- int search (vector< string > &seq, cube node, int price, int bound, int stage)

  Rekurencyjna funkcja przeszukiwania IDA∗.
- vector< string > ida_star (cube root, int stage)

  Implementacja algorytmu IDA∗ dla danej fazy.

Variables

- vector< string > moves
- vector< string > moves_phase1

  Lista wszystkich dozwolonych ruchów w fazie 1 algorytmu.
- vector< string > moves_phase2

  Lista dozwolonych ruchów w fazie 2 algorytmu.
- eoh EOH

  Heurystyka orientacji krawędzi (Edge Orientation Heuristic)
- cph CPH

  Heurystyka pozycji narożników (Corner Permutation Heuristic)
- eph EPH

  Heurystyka pozycji krawędzi (Edge Permutation Heuristic)

6.10.1 Detailed Description

Implementacja algorytmu IDA∗.

Zawiera logikę przeszukiwania przestrzeni stanów w celu znalezienia rozwiązania.

6.10.2 Macro Definition Documentation

6.10.2.1 inf

```
#define inf 1000000000
```

Stała reprezentująca „nieskończoność” używaną w IDA∗.

6.10.3 Typedef Documentation

6.10.3.1 LL

```
typedef long long LL
```

6.10.4 Function Documentation

6.10.4.1 getheuristic()

```
int getheuristic (
            cube & node,
            int stage )
```

Oblicza wartość heurystyki dla danego stanu kostki.

Maksimum z heurystyk narożników i krawędzi w fazie 2

6.10.4.2 ida_star()

```
vector<string> ida_star (
            cube root,
            int stage )
```

Implementacja algorytmu IDA∗ dla danej fazy.

Parameters

root	Stan początkowy kostki
stage	Faza algorytmu (1 lub 2)

Returns

Para: (lista ruchów, głębokość rozwiązania)

6.10.4.3 is_goal_phase1()

```
bool is_goal_phase1 (
            cube & c )
```

Sprawdza, czy kostka spełnia warunki końcowe fazy 1.

Sprawdzenie orientacji narożników

Sprawdzenie orientacji krawędzi

Sprawdzenie, czy krawędzie UD znajdują się w warstwach UD

6.10.4.4 is_goal_phase2()

```
bool is_goal_phase2 (
            cube & c )
```

Sprawdza, czy kostka jest rozwiązana w fazie 2.

Sprawdzenie poprawności narożników

Sprawdzenie poprawności krawędzi

Wszystkie elementy są na właściwych miejscach

6.10.4.5 search()

```
int search (
            vector< string > & seq,
            cube node,
            int price,
            int bound,
            int stage )
```

Rekurencyjna funkcja przeszukiwania IDA∗.

Returns

Najmniejszy przekroczony koszt lub -1 jeśli znaleziono rozwiązanie

6.10.5 Variable Documentation

6.10.5.1 CPH

```
cph CPH
```

Heurystyka pozycji narożników (Corner Permutation Heuristic)

6.10.5.2 EOH

```
eoh EOH
```

Heurystyka orientacji krawędzi (Edge Orientation Heuristic)

6.10.5.3 EPH

```
eph EPH
```

Heurystyka pozycji krawędzi (Edge Permutation Heuristic)

6.10.5.4 moves

```
vector<string> moves
```

6.10.5.5 moves_phase1

```
vector<string> moves_phase1
```

Initial value:
```
= {
"U",
"D",
"R",
"L",
"F",
"B",
}
```

Lista wszystkich dozwolonych ruchów w fazie 1 algorytmu.

Zawiera pełny zestaw obrotów ścian (90°, -90°, 180°).

6.10.5.6 moves_phase2

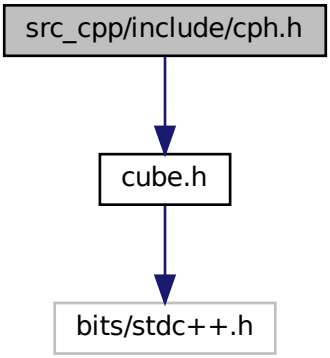
```
vector<string> moves_phase2
```

Initial value:
```
= {
    "U", "Up","U2",
    "D", "Dp","D2",
    "R2","L2","F2","B2"
}
```

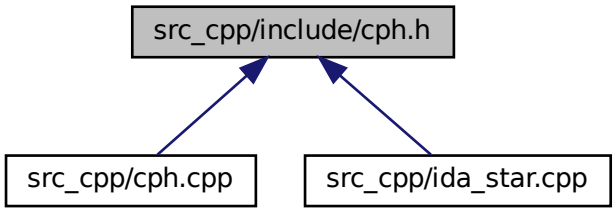
Lista dozwolonych ruchów w fazie 2 algorytmu.

6.11 src_cpp/include/cph.h File Reference

```
#include "cube.h"
```
Include dependency graph for cph.h:



This graph shows which files directly or indirectly include this file:

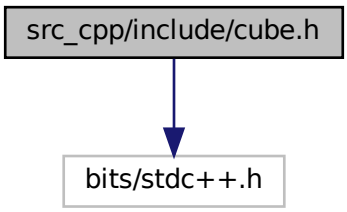


Classes

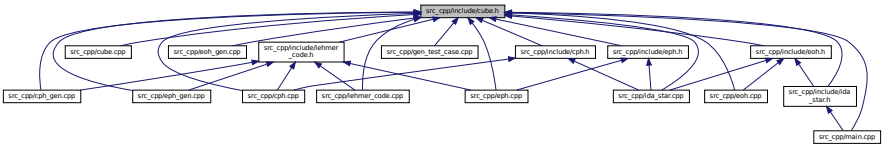
- struct cph

6.12 src_cpp/include/cube.h File Reference

`#include <bits/stdc++.h>`
Include dependency graph for cube.h:



This graph shows which files directly or indirectly include this file:



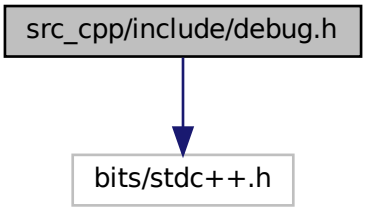
Classes

- class cube

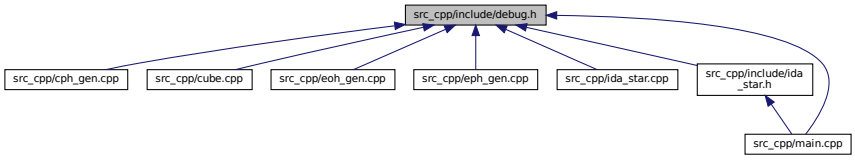
6.13 src_cpp/include/debug.h File Reference

```
#include <bits/stdc++.h>
```

Include dependency graph for debug.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define debug(X...) {}

6.13.1 Macro Definition Documentation

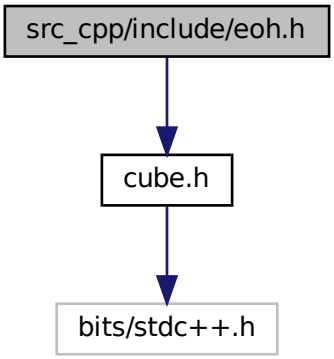
6.13.1.1 debug

```
#define debug(
              X...  ) {}
```

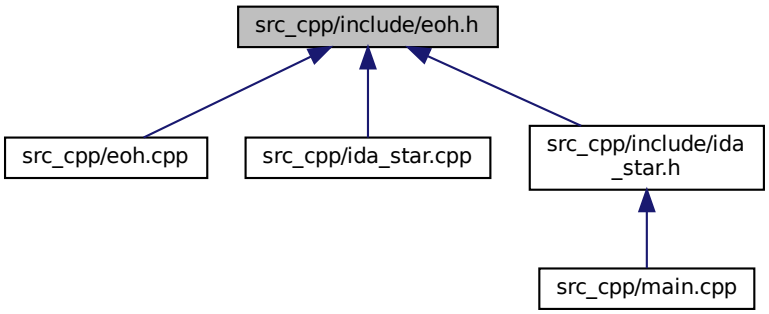
6.14 src_cpp/include/eoh.h File Reference

```
#include "cube.h"
```

Include dependency graph for eoh.h:



This graph shows which files directly or indirectly include this file:



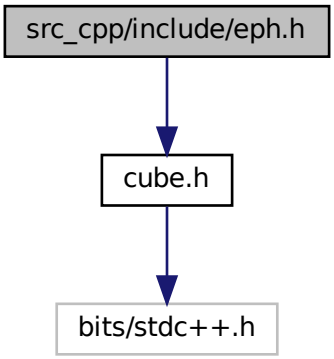
Classes

- struct eoh

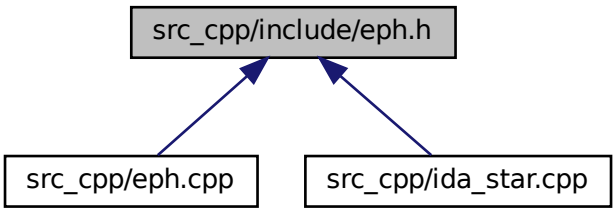
6.15 src_cpp/include/eph.h File Reference

```
#include "cube.h"
```

Include dependency graph for eph.h:



This graph shows which files directly or indirectly include this file:



Classes

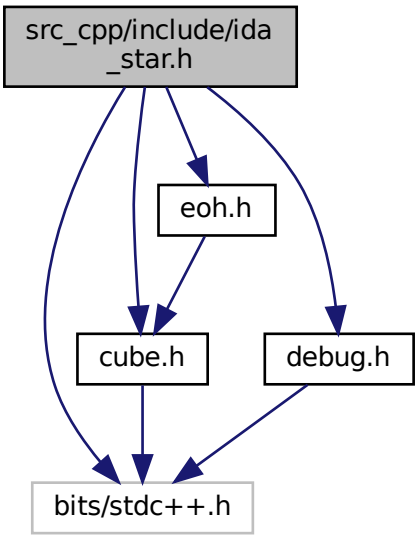
- struct eph

6.16 src_cpp/include/ida_star.h File Reference

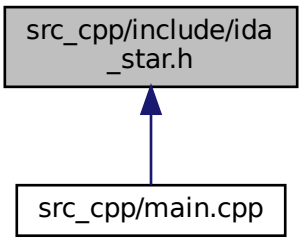
```
#include <bits/stdc++.h>
#include "cube.h"
#include "debug.h"
```

```
#include "eoh.h"
```

Include dependency graph for ida_star.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef long long LL

Functions

- vector< string > ida_star (cube root, int stage)

  Implementacja algorytmu IDA∗ dla danej fazy.

6.16.1 Typedef Documentation

6.16.1.1 LL

```
typedef long long LL
```

6.16.2 Function Documentation

6.16.2.1 ida_star()

```
vector<string> ida_star (
            cube root,
            int stage )
```

Implementacja algorytmu IDA∗ dla danej fazy.

Parameters

root	Stan początkowy kostki
stage	Faza algorytmu (1 lub 2)

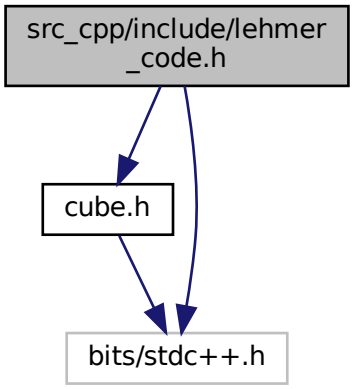
Returns

Para: (lista ruchów, głębokość rozwiązania)

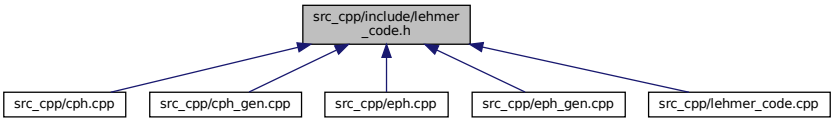
6.17 src_cpp/include/lehmer_code.h File Reference

```
#include "cube.h"
#include <bits/stdc++.h>
```

Include dependency graph for lehmer_code.h:



This graph shows which files directly or indirectly include this file:



Functions

- int lehmer_code (vector< int > &perm)

6.17.1 Function Documentation

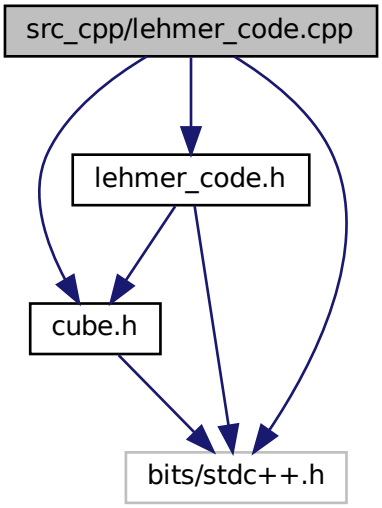
6.17.1.1 lehmer_code()

```
int lehmer_code (
            vector< int > & perm )
```

6.18 src_cpp/lehmer_code.cpp File Reference

```
#include "lehmer_code.h"
#include "cube.h"
#include <bits/stdc++.h>
```

Include dependency graph for lehmer_code.cpp:



Functions

- int lehmer_code (vector< int > &perm)

Variables

- constexpr int fact [12]

6.18.1 Function Documentation

6.18.1.1 lehmer_code()

```
int lehmer_code (
            vector< int > & perm )
```

6.18.2 Variable Documentation

6.18.2.1 fact

```
constexpr int fact[12]  [constexpr]
```

Initial value:

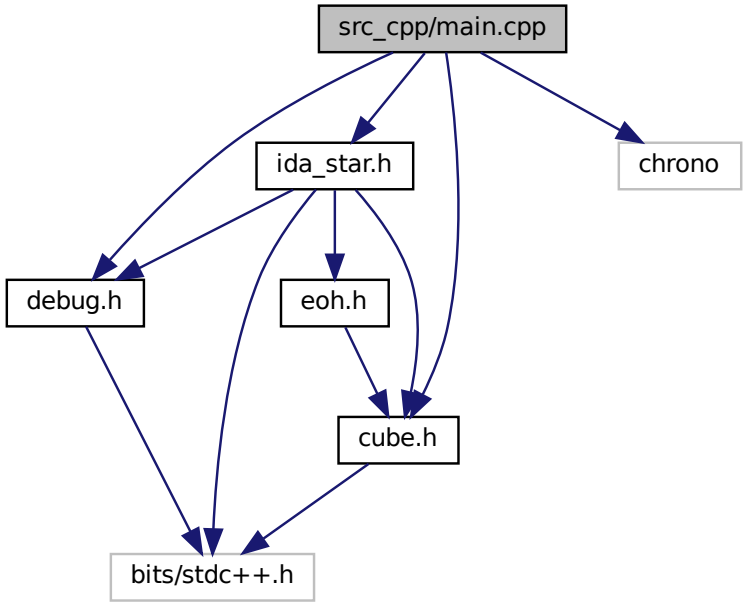
```
= {
    1, 1, 2, 6, 24, 120, 720, 5040,
    40320, 362880, 3628800, 39916800
}
```

6.19 src_cpp/main.cpp File Reference

Plik główny programu solvera.

```
#include "debug.h"
#include "cube.h"
#include "ida_star.h"
#include <chrono>
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

6.19.1 Detailed Description

Plik główny programu solvera.

Obsługuje argumenty wiersza poleceń i steruje procesem rozwiązywania.

6.19.2 Function Documentation

6.19.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

6.20 src_py/cli_handler.py File Reference

Moduł obsługi interfejsu wiersza poleceń (CLI).

Classes

- class cli_handler.SolverCLI

  Klasa zarządzająca komunikacją CLI i integracją.

Namespaces

- cli_handler

Variables

- cli_handler.BASE_DIR = Path(__file__).resolve().parent.parent
- string cli_handler.DATA_DIR = BASE_DIR / "data"
- string cli_handler.INPUT_FILE = DATA_DIR / "cube_state.txt"
- string cli_handler.OUTPUT_FILE = DATA_DIR / "solution_steps.txt"
- list cli_handler.FACES_ORDER = ['U', 'D', 'F', 'B', 'L', 'R']
- dictionary cli_handler.VALID_COLORS = {'W', 'Y', 'G', 'B', 'O', 'R'}
- cli_handler.cli = SolverCLI()

6.20.1 Detailed Description

Moduł obsługi interfejsu wiersza poleceń (CLI).

Odpowiada za pobranie danych od użytkownika, sformatowanie ich do specyfikacji wymaganej przez silnik C++ oraz obsługę procesu.

6.21 src_py/input_gui.py File Reference

Classes

- class input_gui.InputGUI

Namespaces

- input_gui

Variables

- dictionary input_gui.COLORS
- dictionary input_gui.HEX_TO_NAME = {v: k for k, v in COLORS.items()}
- dictionary input_gui.POLISH_NAME
- input_gui.root = tk.Tk()
- input_gui.cube_editor = InputGUI(root)

6.22 src_py/vis_gui.py File Reference

Moduł wizualizacyjny kostki Rubika 3D.

Namespaces

- vis_gui

Functions

- def vis_gui.load_data (filename)

  wczytuje kolejne stany kostki z pliku tekstowego
- def vis_gui.build_cube (cube_state)

  Buduje trójwymiarowy model kostki Rubika.
- def vis_gui.draw_cubie (ax, x, y, z, colours)

  Rysuje pojedynczy element kostki Rubika w przestrzeni 3D.
- def vis_gui.update (frame)

  Aktualizuje pojedynczą klatkę animacji.

Variables

- dictionary vis_gui.colour_map
- def vis_gui.moves = load_data("test.txt")
- vis_gui.fig = plt.figure(figsize=(10,10))
- list vis_gui.axes
- vis_gui.ani = FuncAnimation(fig, update, frames=len(moves), interval=1000)
- vis_gui.writer
- vis_gui.fps

6.22.1 Detailed Description

Moduł wizualizacyjny kostki Rubika 3D.

Moduł odpowiada za generowanie animacji kostki Rubika 3D na podstawie danych wejściowych.

Index