Wydział Elektryczny

POLITECHNIKA WARSZAWSKA

Języki i metody programowania 2

`lava`: Dokumentacja końcowa

Oliwia Pawelec, Jakub Żebrowski

prowadzący zajęcia:

Dr inż. Radosław Roszczyk

27.05.2024, modyfikacja: 10.06.2024

Opis Programu

`lava` to jest program interaktywny znajdujący najkrótszą drogę od punktu wejściowego do punktu wyjściowego przez labirynt wczytany przez użytkownika. Program jest interaktywny, pozwalając na graficzną reprezentację labiryntu i ścieżki przez niego.

Program pozwala na wczytywanie labiryntów w formie tekstowej (z pliku `.txt`) lub binarnej (z pliku `.bin`). Po wczytaniu pozwala on na zmianę położenia punktu wyjściowego oraz wejściowego w labiryncie, lub wyznaczenie przez niego ścieżki. Po wyznaczeniu można ścieżkę zapisać na trzy sposoby:

- Do pliku tekstowego, jako lista kroków do wykonania by przejść labirynt.
- Do pliku binarnego, który będzie posiadał i strukturę labiryntu i ścieżkę w zdefiniowanym poniżej formacie.
- Do pliku graficznego, jako zdjęcie labiryntu.

Założenia projektowe

Program rozwiązuje labirynty, które podlegają następującym kryteriom:

- labirynt jest zakodowany albo w formie tekstowej, albo binarnej - inne nieznane formy kodowań są odrzucane
- wymiary labiryntu (licząc same komórki, bez doliczania ścianek między nimi) spełniają nierówność: $5 \leq x, y \leq 1024$

Formaty plików wyjściowych i wejściowych zostaną omówione w następnej sekcji dokumentacji.

Wykorzystywane pliki

Plik wejściowy: obsługiwane są dwa formaty plików: tekstowy i binarny

1. W formacie **tekstowym**, labirynt powinien występować w formacie, który zawiera definicje punktów reprezentujących odpowiednio:
   - P – punkt wejścia do labiryntu
   - K - punkt wyjścia z labiryntu,
   - X - ściana
   - spacja - miejsce, po którym można się poruszać

   Plik ten składać się powinien z pojedynczych znaków, których położenie x, y reprezentuje faktycznie położenie w labiryncie.

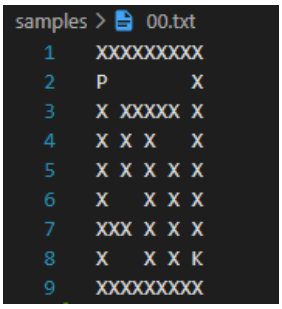

Figure 1: Przykładowy wygląd pliku wejściowego w formacie **tekstowym**

2. W formacie **binarnym**, labirynt powinien wpasowywać się strukturą w narzucony standard, opisany w ostatniej sekcji dokumentacji *Dodatek: struktura specyficznego pliku binarnego*

Plik wyjściowy: zawiera najkrótszą trasę przez wybrany labirynt, opisaną w formacie listy wykonanych kroków, na przykład:

```
START
FORWARD 1
TURNLEFT
FORWARD 4
TURNRIGHT
FORWARD 3
STOP
```

Tu również obsługiwane są oba (**tekstowe** i **binarne**) rodzaje plików. Kodowanie plików zachodzi w analogiczny sposób co do obsługi pliku wsadowego. Jeśli użytkownik zechce rozwiązanie w formie **tekstowej**, do pliku wyjściowego wypisana zostanie lista kroków. Jeśli jednak w formie **binarnej**, do pliku wyjściowego zostanie zapisany zakodowany w formie binarnej labirynt, oraz pod nim zakodowaną (również binarne) listę kroków do przejścia najkrótszej trasy.

W przypadku wyboru zapisu do pliku graficznego, zostanie zrobiony "zrzut ekranu" panelu z labiryntem, który zostanie zapisany do pliku ze wskazanej ścieżki.

Wygląd programu

Po rozpoczęciu programu widoczne jest okno główne, pozwalające na wykonanie wyżej wymienionych funkcji. Składa się ono z:

1. **górnego paska**, który pozwala na wczytywanie oraz zapisywanie labiryntu
2. **głównego panelu**, który umożliwia zobaczenie labiryntu i ścieżki

3. **paska narzędzi**, który pozwala na wywołanie:
    - **Solve** - znajduje najkrótszą ścieżkę w labiryncie,
    - **Set Start** - ustawienie punktu wejściowego w labiryncie,
    - **Set End** - ustawienie punktu wyjściowego w labiryncie,
    - **Clear Path** - Wyczyszczenie wcześniej zaznaczonej ścieżki w labiryncie,
    - **Remove** - Usunięcie widoku labiryntu z głównego panelu.

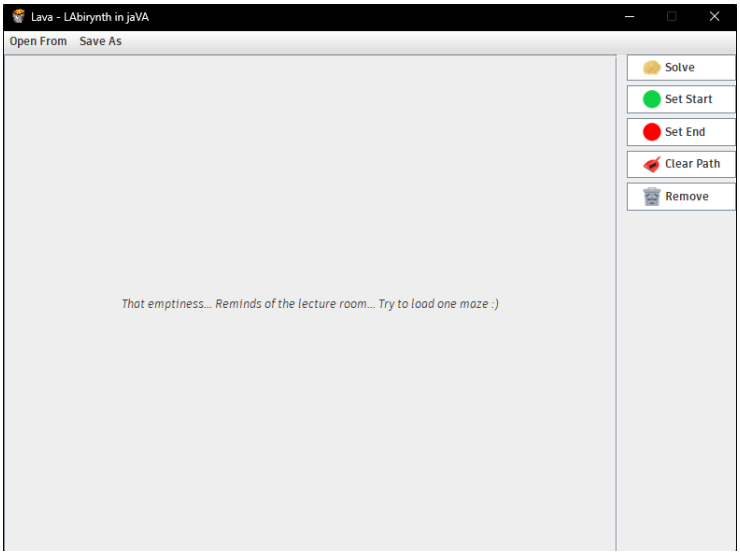

Figure 2: Okno główne i interfejs

Sposób użycia

Wczytywanie plików

Wczytywanie plików można osiągnąć przez wybów opcji *Open File* z górnego paska. Po jej wyborze dalej wybiera się, czy program ma wczytać plik tekstowy czy binarny. Po dalszym wyborze otwiera się okno, które pozwala na wybór pliku z labiryntem w formacie wcześniej wybranym.

W przypadku wybrania pliku, który zawiera w sobie błędy, program nie wczyta labiryntu i poinformuje użytkownika o błędzie. Przykładami błędnego wczytywania jest wczytanie niepoprawnego rodzaju pliku, albo pustego lub niepoprawnie formatowanego pliku.

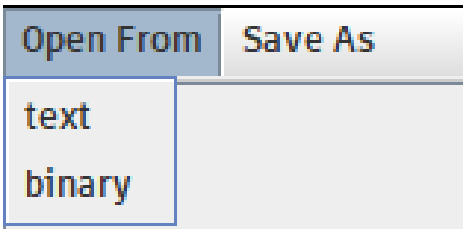

Figure 3: Otwieranie pliku z labiryntem

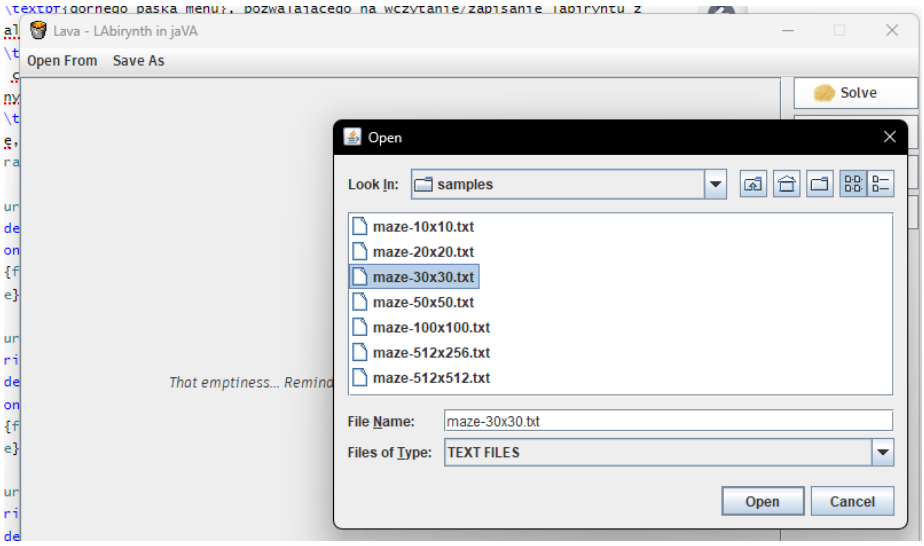

Figure 4: Okno wyboru pliku z labiryntem z kodowaniem tekstowym

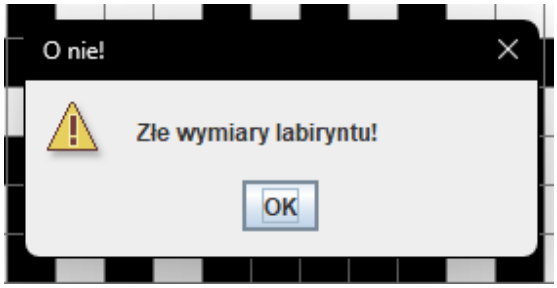

Figure 5: Próba wczytania pustego labiryntu

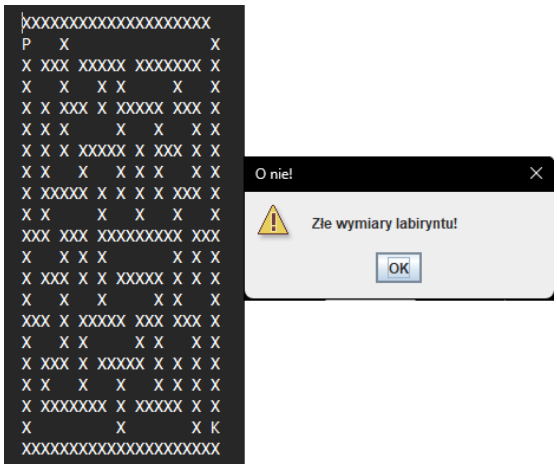

Figure 6: Próba wczytania uszkodzonego labiryntu

Działanie na labiryncie

Po poprawnym wczytaniu pliku labirynt wyświetli się na głównym panelu. Widząc go można też wykonywać na nim działania, czyli zmienić jego punkt wejściowy lub wyjściowy, jak i znaleźć przez niego ścieżkę. Znalezienie ścieżki jest osiągnięte przez naciśnięcie na przycisk `Solve`, co zaznaczy ścieżkę na żółto jeśli istnieje. Aby usunąć zaznaczoną wcześniej ścieżkę, można wybrać opcję `Clear Path`.

Aby zamienić pozycję punktu wejściowego lub wyjściowego, najpierw trzeba nacisnąć na przycisk `Set Start` lub `Set End`, a następnie nacisnąć jedną z kratek na głównym panelu. Wejście jest zaznaczone zieloną kratką, a wyjście czerwoną kratką.

Ostatnia opcja to jest `Remove`, która usuwa labirynt z głównego panelu. Nie usuwa to wczytanego pliku - tylko jego podgląd.

Zapisywanie plików

Zapisywanie plików jest wykonywane przez wybór opcji `Save As` w górnym pasku. Po jej naciśnięciu można wybrać tryb zapisu - Plik tekstowy, plik binarny, lub zdjęcie. Po wybraniu danej opcji można nazwać plik oraz wybrać folder, w którym ma być zapisany. Wybór trybu zdjęcie robi zrzut głównego panelu.

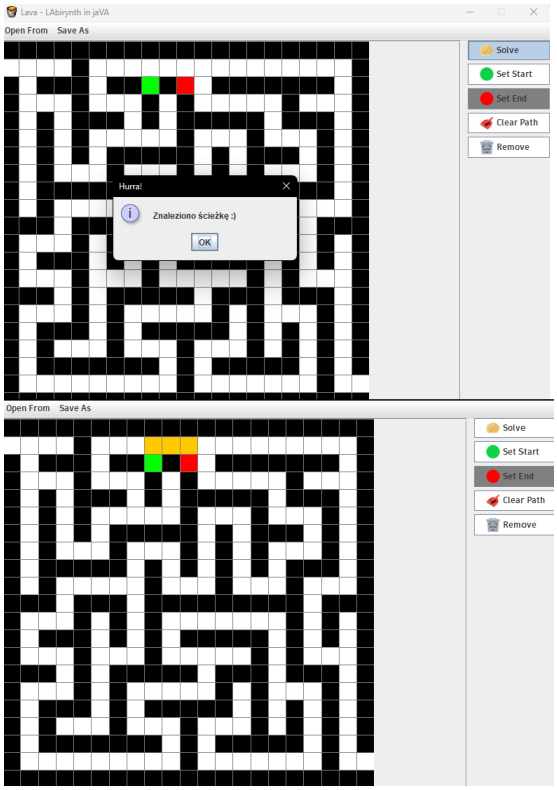

Figure 7: Znalezienie ścieżki przez labirynt

Figure 8: Próba znalezienia nieistniejącej ścieżki

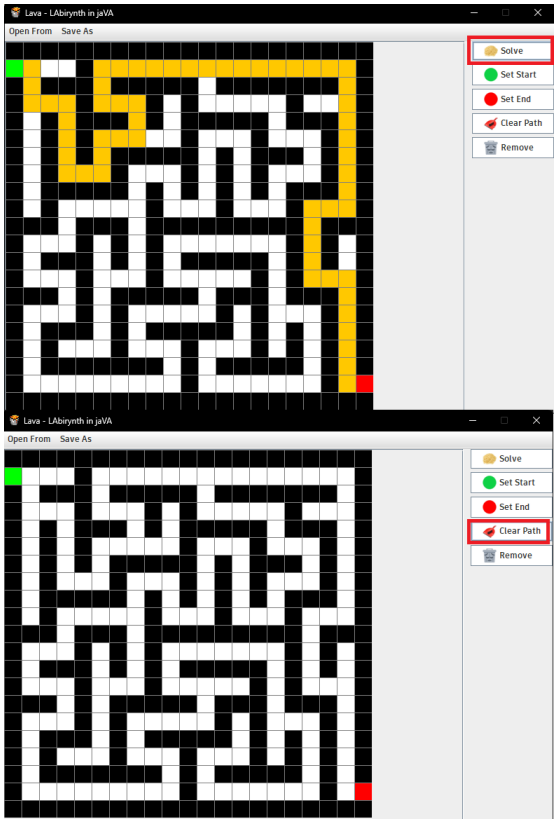

Figure 9: Wyczyszczenie ścieżki

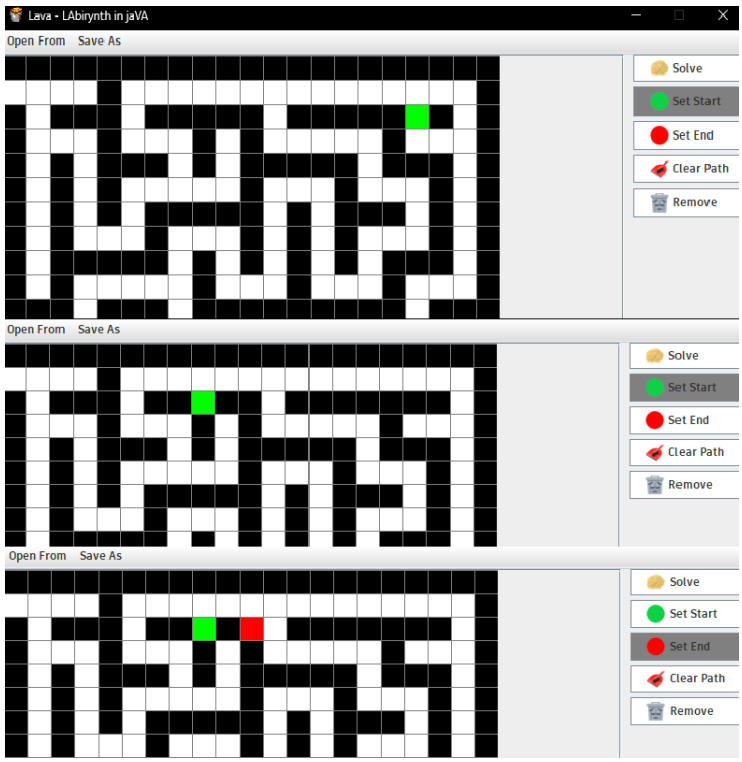

Figure 10: Wybór punktu wejściowego i wyjściowego

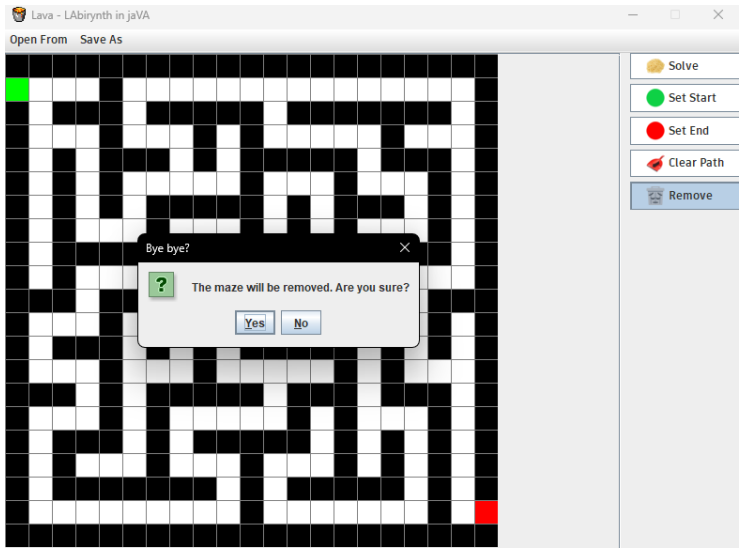

Figure 11: Usunięcie labiryntu

Figure 12: Wybór trybu zapisu

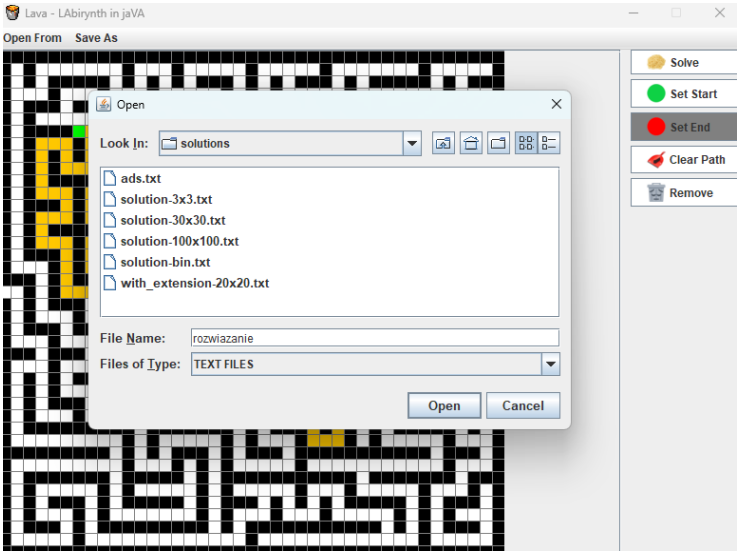

Figure 13: Nazywanie pliku wyjściowego

Struktura programu

Podział na moduły

Najważniejsze moduły programu i ich funkcje:

- Głównego okna
- Wyświetlania (labiryntu)
- Zarządzania (labiryntem)
- Rozwiązania
- Wczytu
- Zapisu

Diagram klas

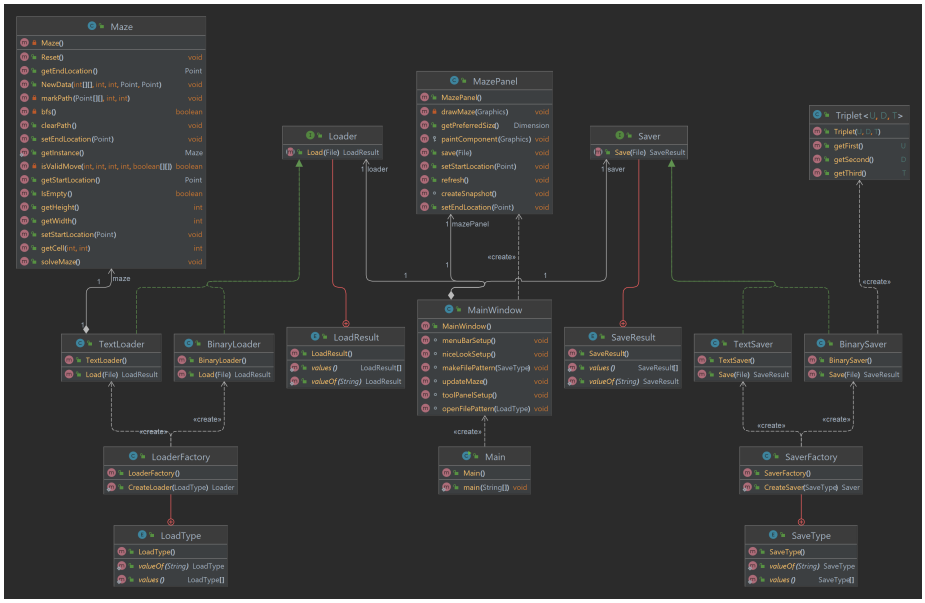

Figure 14: Diagram klas, wygenerowany w środowisku IntelliJ za pomocą pluginu generatora klas UML

Hierarchia klas i plików

- `Main`: punkt wejścia programu
- `MainWindow`: odpowiedzialna za wyświetlanie głównego okna GUI wraz z podstawowymi panelami.

- `MazePanel`: odpowiedzialna za widok panelu z labiryntem - za odświeżanie wczytanego labiryntu, jego rysowanie, nanoszenie na widok punktów wejścia/wyjścia wybranych przez użytkownika oraz wyświetlanie wyprowadzonej ścieżki.
- Obiekty ze słowem `Save`: implementacja wzorca fabryki abstrakcyjnej, zestaw klas służący do zapisu labiryntu do różnych formatów.
- Obiekty ze słowem `Load`: implementacja wzorca fabryki abstrakcyjnej, zestaw klas służący do wczytywania labiryntu z różnych formatów.
- `Maze`: obiekt singleton, przechowujący pojedynczą instancję wczytanego labiryntu. Przechowuje jego strukturę, ewentualnie znalezioną ścieżkę oraz zajmuje się operacjami związanymi z jej wyszukiwaniem.

Szczegóły implementacyjne

Użyte oprogramowanie

Do oprogramowania został użyty język Java (Java Development Kit 17) z biblioteką graficzną Swing.

Algorytm przechodzenia

Do znalezienia najkrótszej ścieżki został użyty algorytm BFS (ang. *Breadth First Search*).

Wywołanie metody `solveMaze()` inicjuje proces rozwiązania, usuwając istniejącą ścieżkę i uruchamiając metodę `bfs()`, która przeszukuje labirynt, oznaczając odwiedzone komórki i sprawdzając możliwe ruchy (prawo, dół, lewo, góra) i dodając je do kolejki do sprawdzenia. Jeśli zostanie znaleziona ścieżka, metoda `markPath()` zaznacza ją w labiryncie, a program wyświetla komunikat o sukcesie, jeśli nie -wyświetlany jest komunikat o niepowodzeniu.

```
public void clearPath() {
...
}

public void solveMaze() {
    // delete all the path from the maze
    clearPath();

    if (bfs()) {
    // jest ścieżka
    } else {
    // nie ma
    }
}
```

```
private boolean bfs() {
    int rows = height;
    int cols = width;
    boolean[][] visited = new boolean[rows][cols];
    //Point[][] parent = new Point[rows][cols];

    Queue<Point> queue = new LinkedList<>();
    queue.add(start);
    visited[start.y][start.x] = true;

    ...

    while (!queue.isEmpty()) {
        Point current = queue.poll();
        ...

        if (currX == end.x && currY == end.y) {
            markPath(parent, currX, currY);
            return true;
        }

        ...
    }

    return false;
}

private void markPath(Point[][] parent, int x, int y) {
    while ( !(x == start.x && y == start.y) ) {
        maze[y][x] = 3;
        ...
    }
}
```

Dodatek: struktura specyficznego pliku binarnego

Dane wejściowe w formacie binarnym podzielone są na 4 główne sekcje:
1. Nagłówek pliku
2. Sekcja kodująca zawierająca powtarzające się słowa kodowe
3. Nagłówek sekcji rozwiązania
4. Sekcja rozwiązania zawierające powtarzające się kroki które należy wykonać aby wyjść z labiryntu

Sekcja 1 i 2 są obowiązkowe i zawsze występują, sekcja 3 oraz 4 są opcjonalne. Występują jeśli wartość pola *Solution Offset* z nagłówka pliku jest różna od 0.

Nagłówek pliku:

Nazwa pola	Wielość w bitach	Opis
File Id	32	Identyfikator pliku: 0x52524243
Escape	8	Znak ESC: 0x1B
Columns	16	Liczba kolumn labiryntu (numerowane od 1)
Lines	16	Liczba wierszy labiryntu (numerowane od 1)
Entry X	16	Współrzędne X wejścia do labiryntu (numerowane od 1)
Entry Y	16	Współrzędne Y wejścia do labiryntu (numerowane od 1)
Exit X	16	Współrzędne X wyjścia z labiryntu (numerowane od 1)
Exit Y	16	Współrzędne Y wyjścia z labiryntu (numerowane od 1)
Reserved	96	Zarezerwowane do przyszłego wykorzystania
Counter	32	Liczba słów kodowych
Solution Offset	32	Offset w pliku do sekcji (3) zawierającej rozwiązanie
Separator	8	słowo definiujące początek słowa kodowego – mniejsze od 0xF0
Wall	8	słowo definiujące ścianę labiryntu
Path	8	słowo definiujące pole po którym można się poruszać
Podsumowanie	**420**	**Sumarycznie nagłówek ma rozmiar 40 bajtów**

Słowa kodowe:

Nazwa pola	Wielość w bitach	Opis
Separator	8	Znacznik początku słowa kodowego
Value	8	Wartość słowa kodowego (Wall / Path)
Count	8	Liczba wystąpień (0 – oznacza jedno wystąpienie)

Sekcja nagłówkowa rozwiązania

Nazwa pola	Wielość w bitach	Opis
Direction	32	Identyfikator sekcji rozwiązania: 0x52524243
Steps	8	Liczba kroków do przejścia (0 – oznacza jeden krok)

Krok rozwiązania:

Nazwa pola	Wielość w bitach	Opis
Direction	8	Kierunek w którym należy się poruszać (N, E, S, W)
Counter	8	Liczba pól do przejścia (0 – oznacza jedno pole)

Pola liczone są bez uwzględnienia pola startowego.