

# Dokumentacja projektu *Labirynt* JIMP2

Norbert Gwiazda i Bartosz Piłat

16.04.2024

## Spis treści:

1	Problem	2
2	Algorytm rozwiązania	3
3	Graficzne przedstawienie działania programu	4
4	Obsługa programu	7
5	Opis implementacji	8

# 1 Problem

W ramach projektu pt. "Labirynt" utworzyliśmy aplikację w języku C, której celem jest rozwiązywanie labiryntów. Program ten czytuje pliki w formie tekstowej lub binarnej, zawierające odpowiadające im reprezentacje labiryntów. Dla plików tekstowych znaki "X" wyznaczają ściany labiryntu, spacje wolną przestrzeń, "P" początek, a "K" koniec. Natomiast dla plików binarnych, początek, koniec, znaki ścian oraz ścieżek opisane są w nagłówku pliku, a ich ułożenie opisane poprzez powtarzające się słowa kodowe. Dla takich plików wczytanych na wejściu, program znajduje ścieżkę będącą rozwiązaniem ich labiryntu.

Oto przykładowy wygenerowany labirynt w formie pliku tekstowego:

```
XXXXXXXXXXXXXXXXXXXXX
P   X   X   X
X X XXX X X X XXX X X
X X X   X X X X X X
X XXX XXX XXX X X XXX
X X   X X   X X X   X
X X XXX XXX X X XXX X
X X X   X   X   X
X X XXX X XXXXXXXXXX
X   X   X X   X X
X XXX XXX X XXX X X X
X X   X   X   X X X
X XXX XXX XXXX X X X
X   X   X   X X X
XXX X X XXXXXXXX X X
X   X X X   X X X X
X XXX X X XXX X X X
X X   X   X X X   X
X XXXXXXXXXX X XXX X
X           X   K
XXXXXXXXXXXXXXXXXXXXX
```

## 2 Algorytm rozwiązania

Program tworzy strukturę znacznika, posiadającą zmienne współrzędnych X i Y oraz zmienną określającą kierunek, w którym jest skierowany. Oś X oznacza kolumny labiryntu, a oś Y rzędy pliku tekstowego zawierającego labirynt.

W celu uruchomienia programu użytkownik wybiera plik określając jego nazwę po fladze "-f", a następnie wybiera tryb programu ("-d" lub "-k").

W przypadku wybrania "-d", należy podać plik w formie tekstowej, a program sam znajdzie rozwiązanie labiryntu.

W przypadku wybrania "-k", należy podać plik w formie binarnej, który program najpierw przetłumaczy na formę tekstową, a następnie znajdzie rozwiązanie zakodowanego labiryntu.

Zasady działania programu są następujące:

- Znacznik rozpoczyna ruch po labiryncie odczytanego z pliku tekstowego zaczynając od znaku P i kończąc na znaku K, przy założeniu, że labirynt posiada rozwiązanie
- Znacznik porusza się stale wzdłuż prawej ściany
- Jeżeli znacznik natrafi na ślepy zaułek, rozpoczyna zalepianie. Zalepianie polega na wstawianiu w labirynt znaków X, aż do napotkania skrzyżowania. Uniemożliwia to ponowne wejście znacznika w dany ślepy zaułek.
- Po dojściu na koniec labiryntu, znacznik stawiany jest z powrotem na początek labiryntu. Robimy to ponieważ skoro znacznikowi udało się dojść na koniec labiryntu, zalepiając po drodze ślepe zaułki, oznacza to że znacznik przechodzący ponownie labirynt, zaczynając od jego początku pójdzie prosto do jego końca.
- W trakcie drugiego przejścia labiryntu, ruchy znacznika zapisywane są do pliku, a na labiryncie wynikowym poszczególne kroki przejścia zaznaczone są przy pomocy znaków "\*".

### 3 Graficzne przedstawienie działania programu



Znacznik zaczyna na początku labiryntu



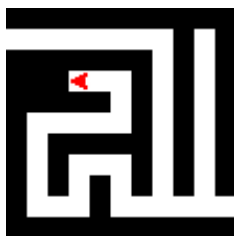
Znacznik porusza się wzdłuż prawej ściany. Nagle napotyka ślepy zaułek.



Znacznik rozpoczyna zalepianie ślepego zaułku.



Znacznik kontynuuje podróż wzdłuż prawej ściany.



Znacznik ponownie napotyka ślepy zaułek.



Zaułek zostaje zalepiony.



Znacznik dochodzi do końca labiryntu!



Znacznik zostaje przeniesiony na początek labiryntu w celu dokonania ponownego przejścia, tym razem wyznaczając trasę rozwiązania.





Znacznik dochodzi do końca, trasa została wyznaczona, a program kończy swoje działanie zapisując wyniki w katalogu "wyniki"

```

XXXXXXXXXXXXX
P*****X X
X XXXXX*X X
X X  X*X X
XXXXX X*X X
X  X*X X
X XXXXX*X X
X X  X*X X
X X X X*X X
X  X  ***K
XXXXXXXXXXXXX

```

7	-	RUCH_W_PRAWO
8	-	RUCH_W_DOL
3	-	RUCH_W_PRAWO

Rozwiązanie labiryntu oraz zapis przejęć utworzone w katalogu "wyniki"

## 4 Obsługa programu

Przed uruchomieniem programu należy go skompilować oraz przygotować do uruchomienia. Obie te czynności zawarte są w pliku Makefile. Plik Makefile uruchamiamy poleceniem *make*. Po uruchomieniu Makefile wszystkie moduły programu są kompilowane, a sam plik wykonywalny zapisywany w ścieżce *bin/labirynt*. Oprócz kompilacji plik Makefile czyści również katalogi *tmp* oraz *wyniki*.

Po skompilowaniu programu możemy go uruchomić. Program uruchamiamy z poziomu katalogu Labirynt, wpisując polecenie *./bin/labirynt*, po którym użytkownik musi podać plik z folderu *dane* zawierający labirynt określając jego nazwę po fladze *-f*, a następnie wybierać tryb programu (*-d* lub *-k*) w zależności od rodzaju podanego pliku (*-d* - tekstowy; *-k* - binarny).

Przykładowe wywołanie programu: *./bin/labirynt -f labirynt*

W przypadku wybrania *-d*, należy podać plik w formie tekstowej. Program po wczytaniu takiego labiryntu podaje jego: nazwę, wybrany tryb programu, czy udało się odczytać plik, liczbę wierszy oraz kolumn labiryntu, umiejscowienie początku oraz końca labiryntu oraz początkową charakterystykę znacznika z miejsca, z którego rozpoczyna się wyszukiwanie rozwiązania.

W przypadku wybrania *-k*, należy podać plik w formie binarnej, który program najpierw przetłumaczy na formę tekstową. Program po odczytaniu takiego labiryntu podaje jego: nazwę, wybrany tryb programu, czy udało się odczytać plik binarny, ID pliku binarnego, znak escape, liczbę kolumn oraz rzędów labiryntu, współrzędne X i Y (liczone od 0) początku oraz końca labiryntu, liczbę słów kodowych, *Solution\_offset* określający czy rozwiązanie labiryntu zawarte jest w pliku binarnym oraz określenie znaków separatora, ściany i ścieżki, które składają się na słowa kodowe. Po zdekodowaniu pliku binarnego użytkownik otrzyma komunikat, czy plik binarny zawiera już rozwiązanie.

Kiedy program zakończy swoje działanie otrzymujemy komunikat: *"Znaleziono rozwiązanie labiryntu"*, jest to potwierdzenie faktu, że program poprawnie odnalazł rozwiązanie podanego labiryntu przez użytkownika.

W celu odczytania wyników należy otworzyć katalog *"wyniki"*. Graficzne rozwiązanie oznaczone znakami *"\*"* zapisane jest w pliku *"sciezka\_rozwiazujaca\_labirynt.txt"*, a zapis kolejnych kroków potrzebnych do rozwiązania labiryntu dla pliku tekstowego zapisany jest w pliku *"zapis\_przejsc.txt"*, natomiast w przypadku pliku binarnego *zapis\_przejsc* określony jest w sekcji *Solution\_offset* poprzez liczbę kroków rozwiązania, na które składa się kierunek (N,E,S,W) oraz liczba pól do przejścia przez znacznik.

## 5 Opis implementacji

Program składa się z 8 modułów:

- labirynt.c
- analiza\_labiryntu.c
- obsluga\_argumentow.c
- odczyt\_labiryntu.c
- odczyt\_pliku\_binarnego.c
- poruszanie\_znacznika.c
- zapis\_trasy.c
- znalezienie\_dowolnego\_przejscia.c

**Moduł labirynt.c** jest rdzeniem programu. To tam spotykają się ze sobą funkcje wszystkich modułów. Funkcja `main` wywołuje moduł odczytu argumentów, na podstawie wyniku, którego określa co dalej powinien robić program. Określa również charakterystykę początkową znacznika i sprawdza, czy ma ona sens wyświetlając komunikat o błędny przypadku skonstruowania labiryntu, jeśli jest on niepoprawny.

**Moduł analiza\_labiryntu.c** odpowiada za określenie liczby kolumn i wierszy pliku tekstowego, określenie punktu startowego i końcowego oraz utworzenie pliku pomocniczego, który będzie edytowany wraz z przejściami wskaźnika.

**Moduł obsluga\_argumentow.c** odpowiada za obsługę odpowiednich flag `"-f"`, `"-d"` oraz `"-k"`. Na podstawie podanych flag odczytuje nazwę podanego pliku oraz zapisuje wybrany przez użytkownika tryb odczytu i podaje go z powrotem do modułu `main`.

**Moduł odczyt\_labiryntu.c** odpowiada za dwie funkcje, umożliwiające kolejno: czytanie bloku przed znacznikiem i bloku, na którym obecnie znajduje się wskaźnik. Funkcje mają zastosowanie w obsłudze ruchu wskaźnika. O ile odczytywanie aktualnego bloku wymaga znajomości jedynie położenia wskaźnika, czytanie bloku przed znacznikiem wymaga znajomości kierunku, w którym porusza się wskaźnik.

**Moduł odczyt\_pliku\_binarnego.c** odcodowuje plik binarny, w przypadku wybrania flagi `"-k"` oraz przypisuje odpowiednim zmiennym wartości. Moduł tłumaczy również plik binarny na plik tekstowy możliwy do odczytania przez człowieka i zapisuje go jako *przetlumaczony\_plik\_binarny*.



**Moduł `poruszanie_znacza.c`** odpowiada za szereg funkcji, które obsługują ruch wskaźnika po pliku labiryntu. Funkcja *ile\_przejsć* sprawdza, czy wskaźnik znajduje się obecnie na skrzyżowaniu, zwykłej drodze, czy w ślepych zaułku. Każdy z tych scenariuszy obsługiwany jest oddzielnie. Funkcja `poruszanie` po labiryncie jest główną funkcją całego modułu. Obsługuje ona przede wszystkim ruch wskaźnika po zwykłej drodze, ale wywołuje również funkcje odpowiedzialne za zalepianie labiryntu. Funkcja `ruch` do przodu odpowiada za poruszanie się wskaźnika do przodu, a funkcja `zmiana_kierunku` zmienia kierunek, jak nazwa wskazuje, zmianę kierunku ruchu wskaźnika. Funkcja `zalepianie` obsługuje zalepianie ślepych zaułków znakami ścian.

**Moduł `zapis_trasy.c`** obsługuje utworzenie pliku wynikowego tj. labiryntu z zaznaczoną ścieżką oraz pliku z kolejnymi przejściami wskaźnika na planszy.

**Moduł `znalezienie_dowolnego_przejscia.c`** przy pomocy funkcji *znalezienie\_dowolnego\_przejscia* obsługuje funkcje związane z tworzeniem pliku pomocniczego oraz wynikowego, jak i określanie parametrów labiryntu oraz znacza, znajdujące się w innych modułach.