📌 1. Nagłówki i funkcje standardowe

#include <stdio.h>

#include <stdlib.h>

Wczytujemy biblioteki do:

- wejścia/wyjścia (printf, scanf, fopen, itp.),
- zarządzania pamięcią (malloc, free).

📌 2. wyswietlLabirynt() – wyświetlanie labiryntu z aktualną pozycją gracza

void wyswietlLabirynt(char *mapa, int n, int px, int py)

- mapa — jednowymiarowa tablica zawierająca znaki labiryntu (# – ściana, spacja – droga).
- n — długość boku labiryntu.
- px, py — współrzędne gracza.

🔁 **Co robi funkcja:**

- Dla każdego wiersza (y) i kolumny (x) wypisuje znak:
  - jeśli (x, y) to pozycja gracza → drukuje "O"
  - w przeciwnym razie → drukuje znak z labiryntu (mapa[i + d*n])

📌 3. sprawdzPole() – sprawdza, czy dane pole jest przejściowe

int sprawdzPole(char *mapa,int n, int x, int y)

✅ **Co robi:**

1. Jeśli pozycja (x,y) jest **poza granicami** – zwraca 0.
2. Sprawdza, czy mapa[y*n + x] to ściana (#, czyli kod ASCII 0x23).
3. Jeśli tak → zwraca 0 (nieprzejezdne).
4. Jeśli nie → zwraca 1 (można wejść).

⚠️ **Błąd logiczny**:
Pętla for(int i = 0; i < n*n; i++) jest **niepotrzebna** — warunek jest niezależny od i. To trzeba usunąć.

📌 4. ruch() – najważniejsza funkcja: logika poruszania się

void ruch(char *mapa, int rozmiar, int kx, int ky)

Parametry:

- mapa — labirynt,
- rozmiar — rozmiar boku,
- kx, ky — kierunek startowy (np. 1, 0 oznacza prawo).

Zmienne:

- px, py — pozycja robota (zaczyna od (0,0)).
- historia — na razie nieużywana tablica (prawdopodobnie do zapisu drogi).
- i — licznik kroków.

🔁 **Główna pętla:**

while(px != rozmiar-1 || py != rozmiar-1)

Dopóki robot nie dojdzie do prawego dolnego rogu (n-1, n-1):

1. usleep(250000) – pauza 0.25 s (dla efektu wizualnego).
2. Wyświetla krok i współrzędne.
3. Wywołuje wyswietlLabirynt() z aktualną pozycją.
4. Sprawdza w którą stronę aktualnie patrzy (np. kx = 0, ky = -1 to góra) i:
    - próbuje skręcić w prawo,
    - jeśli się nie da, idzie prosto,
    - jeśli nie, skręca w lewo,
    - jeśli nie, zawraca.

Ten system opiera się na **algorytmie "prawej ręki"** – zawsze najpierw próbujemy skręcić w prawo.

📌 **5. main() – punkt startowy programu**

int main()

Co robi:

1. Pyta użytkownika o rozmiar (scanf).
2. Alokuje pamięć dla labiryntu (malloc).
3. Otwiera plik 9x9_1.txt i wczytuje mapę.
4. Wywołuje funkcję ruch() z parametrem 1,0, czyli startując w prawo.
5. Kończy działanie.