

eMagazynier — Trasa przez magazyn

Adresat: **wdrożeniowiec (konfiguracja w enowie)**

Wersja dokumentacji: **2026-09-01**

SPIS TREŚCI

1. Po co to jest

2. Jak liczy się trasa

3. Parowanie rzędów

4. Kolejność strefy

5. Cecha TrasaMagazynu

6. Gotowe warianty

6.1 Alejka ślepa

6.2 Hala nieregularna

6.3 Numer na lokalizacji

7. Operacje

8. Pułapki

9. Diagnostyka

1. Po co to jest

Przejdźcie hali to największy pojedynczy koszt kompletacji i odkładania. Kolektor układa więc obie listy — odłożenia i kompletacji — w **kolejności obchodzenia magazynu**.

Pozycje sortuje jedna liczba policzona z adresu lokalizacji: **klucz trasy**. Magazynier idzie od pierwszego przystanku do ostatniego bez zawracania, a kolejność jest zawsze taka sama, więc da się jej nauczyć.

Alternatywą jest porządek, który o hali nie wie nic — nazwa towaru, numer pozycji na dokumencie, rodzaj podpowiedzianej lokalizacji. Każdy z nich ustawia listę wobec magazynu przypadkowo: A-04-01 , A-06-01 , a zaraz po nich A-01-01 , czyli powrót pod regał, przy którym się już stało.

ile to daje w metrach

— hala 6 rzędów × 10 pionów × 3 poziomy, listy po 30 pozycji, 5 losowań (TrasaBrogTests ; model geometrii jest niezależny od wzoru klucza, więc mierzy drogę, a nie sam siebie):

porządek listy	droga	
nazwa towaru — kolejność bez wiedzy o hali	408–514 m	—
naïwny alfabet adresu	118–137 m	lepiej, ale wraca na początek każdej alejki
klucz trasy (serpentina)	91–96 m	~79,7% wobec nazwy towaru

Dруги wiersz jest tu ważniejszy od pierwszego: samo posortowanie po adresie załatwia większość zysku, ale zostawia magazyniera z powrotem na początku każdej alejki. Dopiero serpentina — opisana w następnej sekcji — zbiera te ostatnie ~25%.

Pomiar dotyczy **drogi**, nie czasu pracy: nie uwzględnia sięgania na wyższe półki, szerokości wózka ani ruchu innych magazynierów. Lokalizacja ma w modelu współrzędne w metrach, a magazynier chodzi korytarzami i alejkami.

Trasa jest tak dobra, jak opis hali. Klucz liczy się z adresu lokalizacji (strefa, rząd, pion, poziom), więc lokalizacja bez wypionionego adresu czyta się jako strefa bez numeru i trafia **na koniec** listy, a hala o układzie innym niż regaly parami wymaga ustawienia — patrz [Parowanie rzędów](#) i [Kolejność strefy](#).

2. Jak liczy się trasa

Wzorzec to **serpentina** (S-shape): wchodzisz w alejkę, idziesz do końca i wychodzisz drugą stroną, zamiast wracać po własnych śladach.



W alejkę 1 wchodziśz od pionu 10 — dlatego C-04 stoi przed C-01, i tak ma być.
Człony klucza, od najważniejszego: strefa — alejka — pion — poziom — strona alejki

Kolejność wewnątrz alejki

Kolumna regału jest obsługiwana w całości — obie strony, od podłogi w górę — a dopiero potem idzie się do kolumny dalej. Klucze poniżej są z hali, w której strefa ma **numer 1**:

adres	klucz	co się zmieniło
A-01-00	1 000 000 001 000	start: pierwsza kolumna, podłoga
B-01-00	1 000 000 001 001	naprzeciwko, druga strona alejki (+1)
A-01-01	1 000 000 001 010	półka wyżej, ta sama kolumna (+10)
B-01-01	1 000 000 001 011	znów naprzeciwko
A-02-00	1 000 000 002 000	dopiero teraz kolejna kolumna (+1000)
C-09-00	1 000 019 990 000	następna alejka — wejście od dalekiego końca
C-01-00	1 000 019 998 000	w tej alejce kolumna 01 jest ostatnia

Pierwszy człon to numer strefy, nie jej symbol. Te same adresy w strefie nr 2 dadzą 2 000 000 001 000 , a w strefie bez numeru — 999 000 000 001 000 , czyli koniec trasy. Skąd numer, mów [następna sekcja](#).

3. Parowanie rzędów — pierwsza rzecz do ustawienia

Cecha **danych** na magazynie. Mówi, ile rzędów magazynier obsługuje w jednym przejściu alejką.

gdzie

kartoteka magazynu → cecha Parowanie rzędów (liczba całkowita)

kto zakłada

akcja menu **Magazyn — Kolektory — Czynności — WebAPI — Konfiguracja**

zakres

per magazyn — dwie hale tej samej bazy mogą mieć różny układ

wartość	znaczenie
pusta / 0	2 — wartość domyślna
2	regaly stoją parami naprzeciw siebie: (A,B) to jedna alejka, (C,D) następna
1	każdy rząd ma własny korytarz — osobne przejście
3 i więcej	tylko rzędów na alejkę; układ nietypowy, ale wzór to przyjmie

Ta cecha nie służy do optymalizacji, tylko do niepsucia trasy. Zmierzone tym samym modelem geometrii:

hala	klucz 2	klucz 1
regaly parami	24,0 m	24,0 m
osobne korytarze	350,4 m	187,2 m

Ustawione zgodnie z halą nie zyskujesz nic. Ustawione niezgodnie tracisz **+87% drogi**. Dlatego przy rzędach pojedynczych wpisanie **1** jest obowiązkiem wdrożeniowym, a nie strojeniem.

Jeżeli hala nie daje się opisać *żadną* jedną liczbą (A+B parami, C na drugim końcu, D w poprzek pod ścianą) — parowanie nie wystarczy. Przejdź do [wariantu z mapą rzędów](#).

4. Kolejność strefy — druga rzecz do ustawienia

Cecha **danych** na elemencie słownika. Mówi, którą strefę magazynier obchodzi pierwszą — i którą zapelnia pierwszą.

gdzie

element słownika **Strefa** → cecha Kolejność strefy (liczba całkowita)

kto zakłada

akcja menu **Magazyn — Kolektory — Czynności — WebAPI — Konfiguracja**

zakres

per strefa — numerujesz od **1**, w kolejności obchodzenia

wartość	trasa	odkładanie
1, 2, 3...	kolejność obchodzenia — 1 pierwsza	zapelniasz w tej samej kolejności
pusta / 0	strefa idzie na koniec trasy	nie jest proponowana do odkładania

Pusta wartość znaczy dwie rzeczy naraz, bo obie wynikają z jednego faktu: „ta strefa nie jest miejscem składowania”. Na trasę wchodzi mimo to — może tam coś leżeć i trzeba po to dojść — ale sugestia odłożenia jej nie zaproponuje. Typowe ustawienie: hala **1**, namiot na gabaryty **2**, a przyjęcie / kontrola / wydanie **bez wartości**.

Konfiguracja zakłada strefę M (Magazynowania) i nadaje jej numer 1 — na tym jej rola się kończy. Każdą kolejną strefę zakładasz Ty i Ty wpisujesz jej numer, jeśli klient tego potrzebuje. Pamiętaj tylko, co znaczy brak numeru: strefa nie rozdziela trasy i nie jest proponowana do odkładania. Drugiemu miejscu składowania (namiot, antresola) nadaj więc numer **zanim** ruszy pierwsze odkładanie.

Numerem strefy ustawia się też „zapelnij halę, dopiero potem namiot”. Sugestia bierze pierwszego kandydata, który się mieści, w kolejności listy — a listę porządkuje właśnie ten numer. Priorytet zapelniania jest więc liczbą, nie regułą do napisania.

Zdjęcie numeru uniemożliwia także stałą lokalizację towaru. Towar, który ma przypisaną stałą lokalizację w strefie bez numeru, **nie pojedzie tam** — sugestia wraca do regału, a w dzienniku modułu zostaje ślad z nazwą towaru i gniazda. Przypisanie na kartotece **nie ruszamy**: gdy strefa wróci na trasę, stała zadziała znowu sama.

Numer strefy jest stwierdzeniem o hali, a stała — intencją wobec jednego towaru: przy zamykaniu strefy wygrywa hala, bo intencja powstaje, gdy hala wyglądała inaczej. Stała ustępuje tu tak samo, jak ustępuje od zawsze, gdy towar się w niej nie mieści.

W bazie, w której nikt nie numerował stref, nic się nie zmienia — reguła włącza się dopiero wtedy, gdy numeracja jest faktycznie używana. Nie dotyczy też **pobierania**: towar leżący w strefie bez numeru nadal trzeba stamtąd zabrać, więc kompletacja go widzi (idzie tylko na koniec trasy).

5. Cecha TrasaMagazynu

Cecha **algorytmiczna** — sięgasz po nią dopiero wtedy, gdy parowanie rzędów i kolejność stref nie wystarczają.

gdzie

Narzędzia → Opcje → Systemowe → Definicje cech, tabela CfgNodes , zakładka Algorytm

kto zakłada

akcja menu **Magazyn — Kolektory — Czynności — WebAPI — Cecha trasy magazynu**

typ

Liczba rzeczywista (patrz [pułapki](#))

zakres

globalny — ten sam kod dla każdego magazynu, rozdzieli je po `magazyn.Symbol`

Po założeniu **nic się nie zmienia**: wstawiony algorytm woła wzór dodatku, więc trasa zostaje ta sama. Zmienia się tylko to, kto ją liczy — od tej chwili kod z bazy, nie z DLL.

```
public double Get_Feature_TrasaMagazynu(
    Soneta.Hanglow.Magazyn magazyn,
    Soneta.Handel.DokumentHandlow lokalizacja,
    string operacja,
    string strefa, string rzad, int pion, int poziom)
{
    // ...
}
```

Mniejsza liczba = wcześniej na trasie. **Każda zwrócona wartość jest odpowiedzią — także 0** (to legalny klucz pierwszego przystanku, a nie „policz to za mnie”).

Powrót do wzoru dodatku = skasowanie definicji cechy. Samo wyczyszczenie ciała metody da trasę zerową dla wszystkich lokalizacji, a nie regułę wbudowaną.

Po każdej zmianie ciała algorytmu wymusi rekompilację: Opcje → Systemowe → Kompilator → przycisk „Wymuś rekompilację kodu”.

Skompilowany algorytm siedzi w pamięci **procesu**. Sesje WebAPI są per zapytanie, więc wszystko niżej odwieśa się samo — ale nie ten cache.

Rekompilacja uniemożliwia algorytm **w wszystkich procesach naraz** i mija też cache dyskowy. Kosztuje jednorazowo **około sekundy** — i na tę chwilę czekają wszystkie równoległe zapytania, więc nie rób tego w szczycie kompletacji.

Założenie i skasowanie cechy działa natychmiast, bez rekompilacji — dotyczy ona wyłącznie zmiany ciała.

Kolejne wywołanie akcji Konfiguracja nie nadpisze cechy — Twój kod jest nieetykalny także wtedy, gdy ktoś kliknie ją drugi raz. Skutek uboczny: po aktualizacji dodatku nowe przykłady w komentarzu **nie dojdą** do już założonej cechy. Żeby je zobaczyć, skasuj definicję i załóż ponownie.

6. Gotowe warianty

Wszystkie siedzą w komentarzu wstawianym do cechy przy zakładaniu. Poniżej te, po które sięga się najczęściej.

6.1 Alejka ślepa — zawsze w jedną stronę



Wchodziśz i wychodzisz tym samym końcem, więc pion zawsze rośnie — odwracanie kierunku tylko wydłużałoby drogę. Klucz nie ma więc członu „strona” ani serpentyny: nrRzedu = pion — poziom.

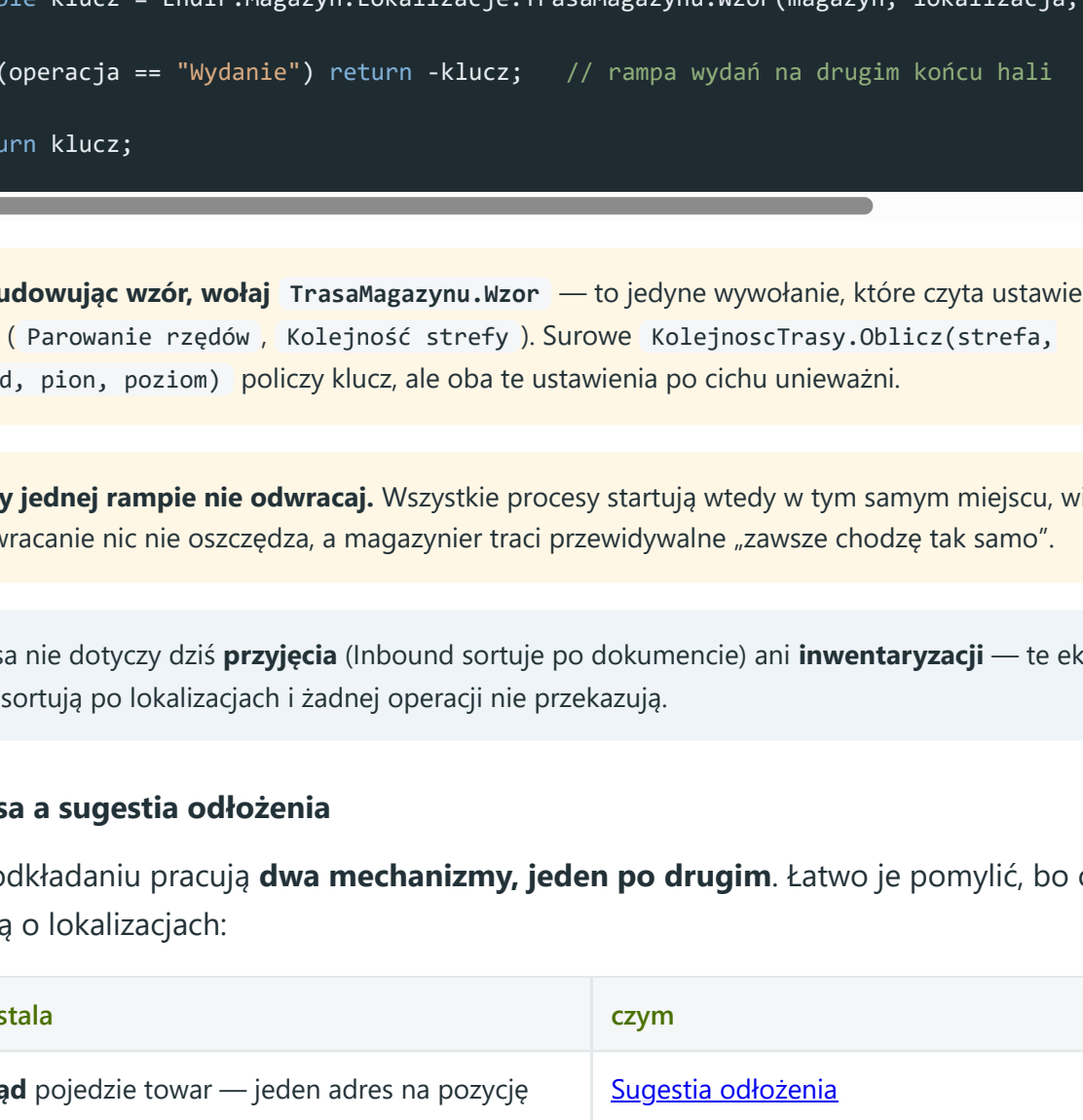
Gdy alejka ma jedno wejście i wraca się nią samą, serpentina nie ma sensu:

```
int nrRzedu = string.IsNullOrEmpty(rzad) ? 1 : char.ToUpper(rzad[0]) - 'A' + 1;
return nrRzedu * 1000000.0 + pion * 1000.0 + poziom * 10.0;
```

Ten wariant liczy klucz sam, bez wzoru dodatku — a ustawienia hali czyta właśnie wzór, więc Parowanie rzędów i Kolejność strefy przestają wpływać na trasę. Powyższy kod nie patrzy na argument strefa , więc cała hala liczy się jak jedna: lokalizację z namiotu przeplotą się z halowymi. Przy więcej niż jednej strefie składowania rozdzieli je **w kodzie** — członem starszym niż nrRzedu , numer wpisany w cesze sam tu nie dojdzie.

Odkładania to nie dotyczy. Numer w cesze Kolejność strefy dalej rozstrzyga, które strefy sugestia w ogóle proponuje i którą zapelnia najpierw — ta ścieżka Twojego algorytmu nie woła.

6.2 Hala nieregularna — mapa rzędów



Kolejność obchodzenia: alejka 0 (A+B) — 1 (D) — 2 (C). Alfabetyzuj C, rząd 0 — trasa mówi D przed C. Rząd 0: alejka 0, nie litera rzędu, strona trzyma A i B w jednym przejściu, kierunek odwraca pion w rzędzie C.

Gdy alejka „przestaje być funkcją litery rzędu: **A+B stoją parami, D w poprzek pod ścianą, C na drugim końcu hali**. Wypisujesz układ wprost:

```
int alejka, strona, kierunek;

switch (string.IsNullOrEmpty(rzad) ? '' : char.ToUpper(rzad[0]))
{
    case 'A': alejka = 0; strona = 0; kierunek = 1; break; // alejka A+B
    case 'B': alejka = 0; strona = 1; kierunek = 1; break; // TA SAMA alejka, pod śc
    case 'D': alejka = 1; strona = 0; kierunek = 1; break; // regał w poprzek, pod s
    case 'C': alejka = 2; strona = 0; kierunek = -1; break; // drugi koniec hali — wr
    default: alejka = 8; strona = 0; kierunek = 1; break; // nowy rząd: na KONIEC tr
}

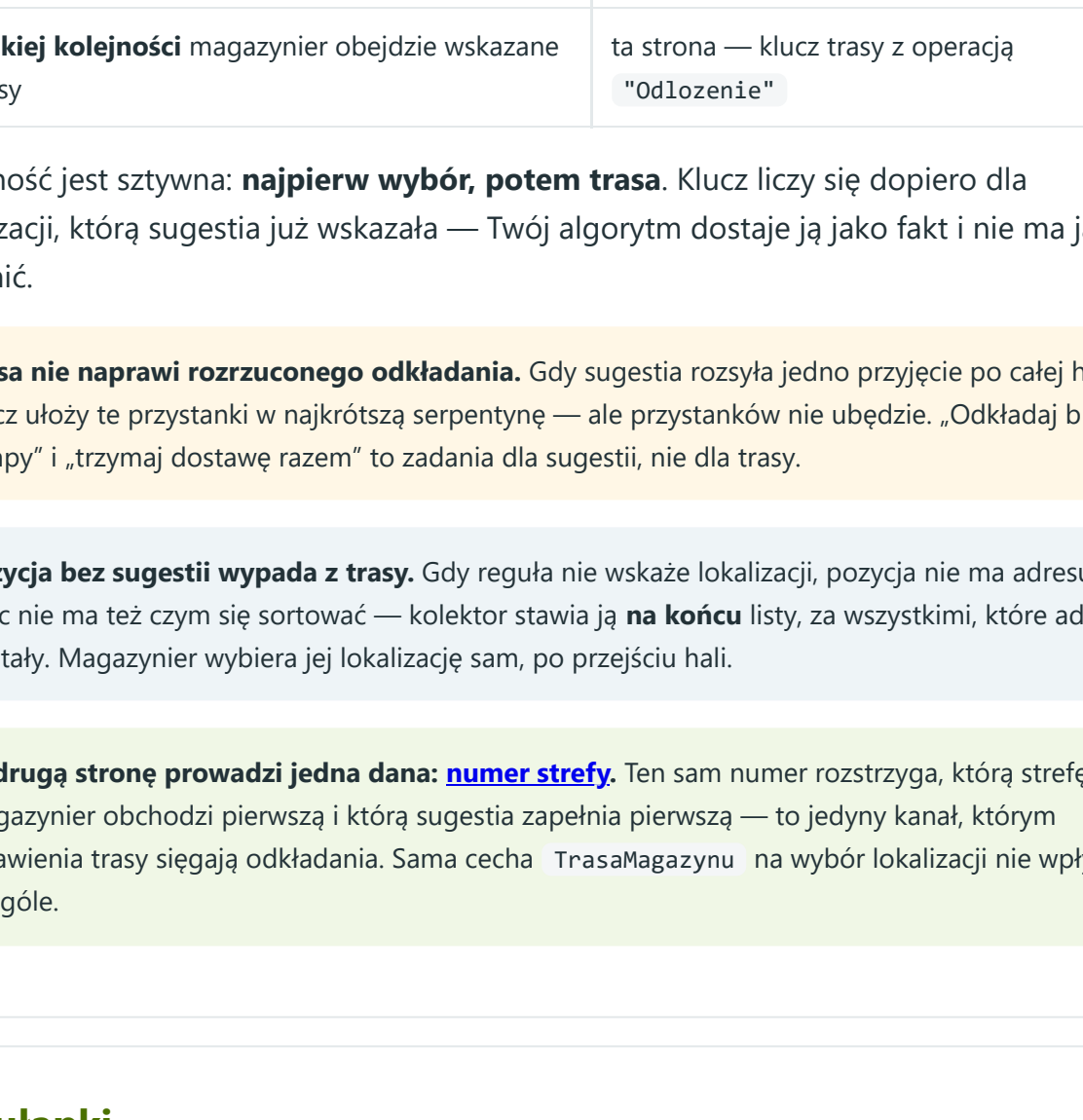
double pionTrasa = kierunek > 0 ? pion : 9999 - pion;
return alejka * 1000000.0 + pionTrasa * 1000.0 + poziom * 10.0 + strona;
```

Trzy rzeczy robią tu całą robotę: **alejka** ustala kolejność obchodzenia (D przed C, choć alfabet twierdzi inaczej), **strona** trzyma A i B w jednym przejściu (ten sam alejka , różna strona → przeplotają się po pion), a **kierunek** = -1 to ręczna serpentina dla regału, którym wracasz ku rampie.

default na końcu (8), nie na początku. Gdy ktoś doda kiedyś rząd E i zapomni o cesze, ma wyglądać za wszystkim, a nie rozzerwać trasę w środku.

Ten wariant też liczy klucz sam, więc Parowanie rzędów i Kolejność strefy przestają wpływać na trasę — ale opisujesz tutaj jej walory, ze strefami włącznie. Na odkładanie numer strefy działało: to osobna ścieżka, która Twojego algorytmu nie woła.

6.3 Nieregularność wewnątrz rzędu — numer na lokalizacji



Numerujesz z przeskakiem: 10, 20, 30... — regał dostawiony później — wchodzi w lukę, nie nie przenumerowujesz — miejsce wyznacza wzór dodatku — mapę da się budować stopniowo.

Gdy numer mapa rzędów nie wystarcza (przeważa w numeracji pionów, regał dostawiony bokiem), zostaje sposób dużych WMS-ów: **Pick Seq** sequence wpisany ręcznie na lokalizacji. Załóż cechę liczbową **PickSeq** na tabeli DokHandlowe (dokument LOK to lokalizacja):

```
if (lokalizacja != null)
{
    var seq = lokalizacja.Features["PickSeq"];

    if (seq != null && System.Convert.ToDouble(seq) > 0)
        return System.Convert.ToDouble(seq);
}

return EndIf_Magazyn.Lokalizacje.TrasaMagazynu.Wzor(magazyn, lokalizacja, strefa, rzad,
```

Numeruj z **przeskokiem** (10, 20, 30...) — wciśnięcie nowej lokalizacji między istniejące nie wymaga wtedy przenumerowania całej hali. Lokalizacje bez **PickSeq** spadają do wzoru, więc mapę można budować stopniowo.

Strefy na koniec tu nie pisz. Odsunięcie antresoli czy namiotu za resztę hali robi się [numerem strefy](#) — bez kodu.

7. Operacje — kto pyta o trasę

Argument **operacja** pozwala dać każdemu procesowi własny kierunek.

wartość	skąd przychodzi
"Odłożenie"	lista odłożenia po przyjęciu (PZ, MMP)
"Wydanie"	kompletacja wydania — lista kompletacyjna. WZ
"PrzesunięcieMM"	kompletacja przesunięcia międzymagazynowego (MMW)
"Lokalizacje"	pasek lokalizacji z zasobem na karcie towaru

Wydanie i przesunięcie są rozdzielone, mimo że kompletują tym samym kodem. To podział procesowy, nie techniczny: towar z WZ jedzie na rampę wydać, a z MMW na rampę przesunąć — u części klientów są to dwa różne końce hali.

```
double klucz = EndIf_Magazyn.Lokalizacje.TrasaMagazynu.Wzor(magazyn, lokalizacja, strefa, rzad, operacja);
if (operacja == "Wydanie") return -klucz; // rampa wydań na drugim końcu hali
return klucz;
```

Obudowując wzór, wołaj TrasaMagazynu.Wzor — to jedyną wywołanie, którego czyta ustawienia hali (Parowanie rzędów, Kolejność strefy). Surowe KolejnośćTrasy,Oblicz(strefa, rząd, pion, poziom) policzy klucz, ale oba te ustawienia po cichu uniemożliwiają.

Przy jednej rampie nie odwracaj. Wszystkie procesy startują wtedy w tym samym miejscu, więc odwracanie nic nie oszczędza, a magazynier traci przewidywalność „zawsze chodzę tak samo”.

Trasa nie dotyczy dziś **przyjęcia** (Inbound sortuje po dokumencie) ani **inwentaryzacji** — te ekrany nie sortują po lokalizacjach i żadnej operacji nie przekazują.

Trasa a sugestia odłożenia

Przy odkładaniu pracują **dwa mechanizmy, jeden po drugim**. Łatwo je pomylić, bo oba mówią o lokalizacjach:

co ustala	czym
Dokąd pojedzie towar — jeden adres na pozycję	Sugestia odłożenia
W jakiej kolejności magazynier obejdzie wskazane adresy	ta strona — klucz trasy z operacją "Odłożenie"

Kolejność jest sztywna: **najpierw wybór, potem trasa**. Klucz liczy się dopiero dla lokalizacji, którą sugestia już wskazała — Twój algorytm dostaje ją jako fakt i nie ma jak jej zmienić.

Trasa nie naprawi rozrzuconego odkładania. Gdy sugestia rozyla jedno przyjęcie po całej hali, klucz ułoży te przystanki w najkrótszą serpentyne — ale przystanków nie ubędzie. „Odkładaj bliżej rampy” i „trzymaj dostawę razem” to zadania dla sugestii, nie dla trasy.

Pozycja bez sugestii wypada z trasy. Gdy reguła nie wskaże lokalizacji, pozycja nie ma adresu, więc nie ma też czym się sortować — nie błądem, tylko cichą degradacją do wzoru wbudowanego, ze śladem w dzienniku. Nieużywane argumenty po prostu zignoruj.

W drugą stronę prowadzi jedna dalsza: numer strefy. Ten sam numer rozstrzyga, którą strefę magazynier obchodzi pierwszą i którą sugestia zapelnia pierwszą — to jedyny kanał, którym ustawienia trasy sięgają odkładania. Sama cecha TrasaMagazynu na wybór lokalizacji nie wpływa w ogóle.

8. Pułapki

Granica dokładności własnego wzoru

Cecha algorytmiczna nie ma w Sonecie typu długiego (FeatureTypeNumber zna Int i Double , nie Long), a klucz mnoży numer strefy przez 10¹². Int kończy się na 2 147 483 647, więc nie mieści nawet samego członu strefy — stąd **Liczba rzeczywista**. Ale i ona liczy całkowite co do strefy tylko do 2⁵³ (9,007 · 10¹⁵).

Wzoru dodatku ta granica nie dotyczy — z konstrukcji. Pierwszym członem klucza jest numer strefy (1–998, brak numeru = 999), więc najwyższy możliwy klucz to 9,99 · 10¹⁴, czyli rząd wielkości pod granicą. Dokładnie dlatego warunkim „koniec trasy” ma 999, a nie 9999.

Granica wraca, gdy klucz liczysz WŁASNYM wzorem — i skutek jest wtedy **częściowy, przez to podstępny**: kolejność stref, alejek i pionów zostaje wierna, ale poziom i strona alejki mogą się skleić, więc trasa zamienia półki w obrębie jednej kolumny i wygląda to na kaprys programu. Dodatek pisze o przekroczeniu w dzienniku, raz na proces. **Lekarstwo: zwracaj mniejszy klucz** albo obuduj TrasaMagazynu.Wzor , którego wynik mieści się zawsze.

Liczba argumentów musi zgadzać się co do jednego

FeatureDefinition, CheckKangs porównuje ją z sygnaturą metody, więc **zostaw sygnaturę dokładnie taką, jaką wstawiaś akcja**. Skreślenie argumentu, którego nie używasz (np. lokalizacja albo operacja), tak samo jak dopisanie własnego, sprawi że cecha **przestanie działać** — nie błędem, tylko cichą degradacją do wzoru wbudowanego, ze śladem w dzienniku. Nieużywane argumenty po prostu zignoruj.

null jest bezpieczny: CheckKangs przepuszcza go dla typów referencyjnych. Lokalizacja bez strefy (strefa == null) ani wywołanie bez dokumentu niczego nie wywracają — ale **sprawdź** lokalizacja != null , zanim sięgniesz po jego cechy.

eMagazynier — dokumentacja wdrożeniowa: trasa przez magazyn • powiązane: [Sugestia odłożenia](#), [Odłożenie na kolektorze](#), [strona produktu](#)