Agnieszka Płoszaj
218353

Problem ucztujących filozofów

Program został zaimplementowany w języku Python, przy pomocy biblioteki *threading*.

Metoda rozwiązania problemu: hierarchia zasobów.

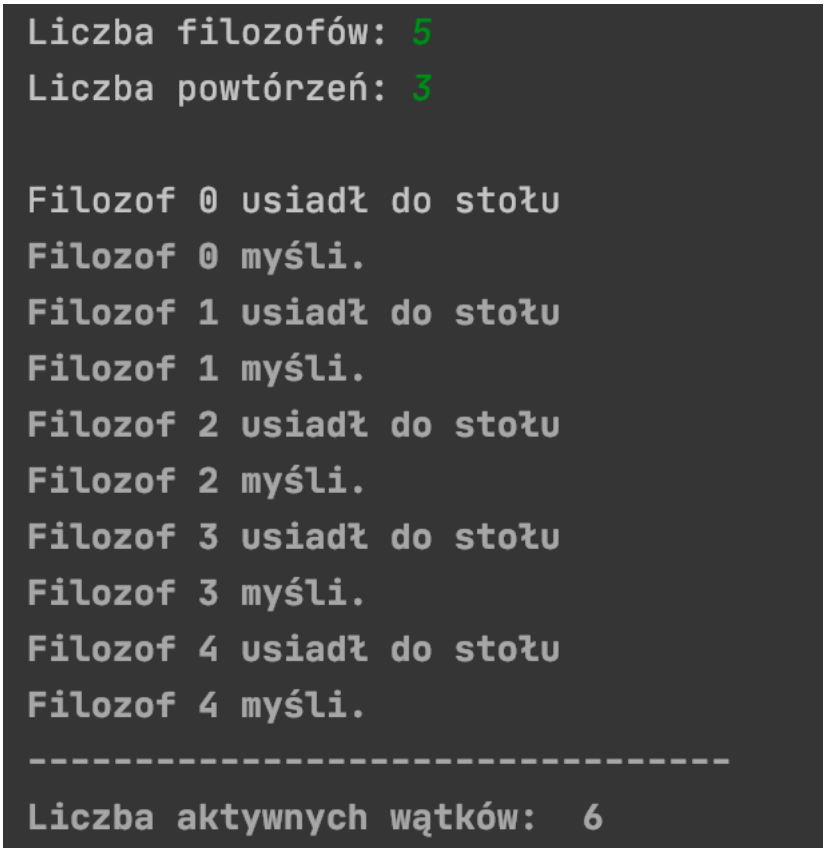
Na początek, program pobiera od użytkownika liczbę filozofów *n* oraz liczbę powtórzeń *repeat* (ile razy ma się wykonać „cykl życia" każdego wątku).
Następnie są tworzone instancje zmiennej warunkowej widelców w liczbie odpowiadającej liczbie filozofów oraz instancje klasy filozofa z pięcioma argumentami (nazwa, numer, lewy widelec, prawy widelec oraz liczba powtórzeń).
W następnej kolejności program przy pomocy metody *start()* rozpoczyna działanie każdego z wątków. Dodatkowo, w programie wykorzystano funkcję *active.Count()*, która zwraca liczbę aktywnych wątków.

Najważniejsze funkcje klasy Philosopher:

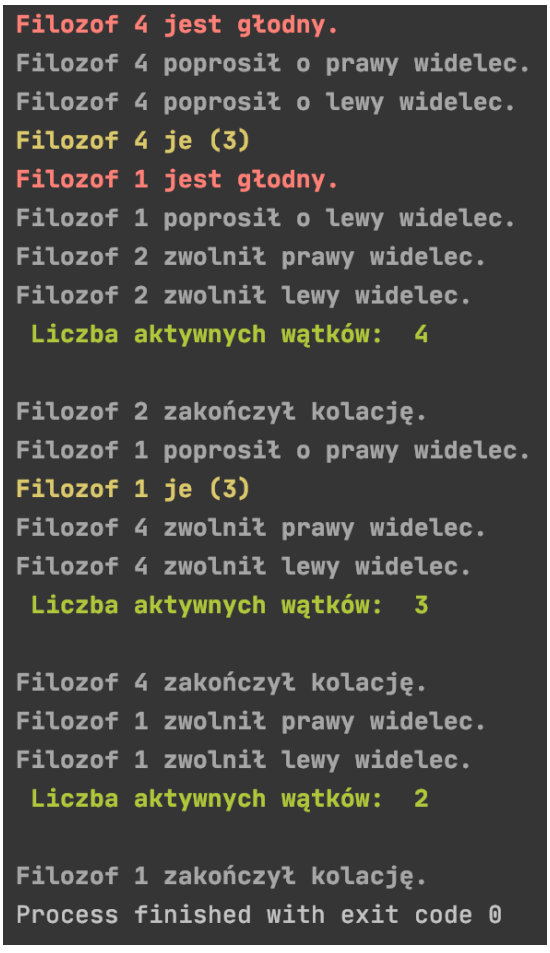
- *life_cycle()* – dopóki aktualna liczba powtórzeń (*end*) jest mniejsza od zadanej liczby powtórzeń (*repeat*), wywołuje kolejno funkcję myślenia (*think()*) oraz głodu (*hunger()*). W przeciwnym razie wywołuje funkcję leave(), która kończy działanie wątku.
- *hunger()* – **odpowiada za hierarchię zasobów**: dla każdego filozofa, za wyjątkiem filozofa o najwyższym numerze, prosi o lewy widelec (który jest widelcem o niższym numerze), a następnie o prawy. Natomiast dla filozofa o najwyższym numerze następuje zamiana kolejności ubiegania się o widelce: najpierw prosi o prawy, a dopiero później o lewy. W konsekwencji, nigdy nie dojdzie do sytuacji, kiedy każdy z filozofów trzyma jeden widelec (rozwiązany został problem zakleszczenia). Dostęp do zmiennych warunkowych w postaci widelców został rozwiązany poprzez metodę ***acquire()***, która wywołuję tę metodę na zamku, dzięki czemu można zablokować zamek, o ile nie został on wcześniej zablokowany (uniemożliwia to korzystanie z konkretnego widelca przez 2 wątki jednocześnie). Jeśli filozofowi uda się uzyskać oba widelce, zostaje wywołana funkcja *eat()*.
- *eat()* – na początku następuje iteracja liczby spożytych posiłków, co umożliwia kontrolowanie, czy nie dochodzi do zagłodzenia. Następnie zostaje wywołana metoda ***release()***. Zwalnia ona zamek, który został uprzednio zablokowany poprzez metodę *acquire()*. W ten sposób, w chwili kiedy dany filozof kończy posiłek, odkłada widelce, dzięki czemu inny filozof, czekający na posiłek może je podnieść. Na koniec następuje powrót do funkcji life_cycle(), gdzie wywoływana jest najpierw funkcja myślenia, dzięki czemu, nie powinno dochodzić do zagłodzenia.

Ilustracja 1 prezentuje pobranie liczby filozofów oraz powtórzeń od użytkownika, następnie inicjalizacji wszystkich wątków oraz ich liczbę.

Ilustracja 2 przedstawia działanie programu oraz poprawne usuwanie wątków.



Ilustracja 1 Inicjalizacja



Ilustracja 2 Działanie programu