

PROGRAMOWANIE W PROLOGU NA ZAJĘCIACH FAKULTATYWNYCH W SZKOLE PONADPODSTAWOWEJ

Mirosława Kołowska-Gawiejnowicz

Wydział Matematyki i Informatyki

Uniwersytet im. Adama Mickiewicza w Poznaniu

ul. Uniwersytetu Poznańskiego 4, 61-614 Poznań

Marika Maćkowiak

I Liceum Ogólnokształcące z Oddziałami Dwujęzycznymi

im. Tomasza Zana we Wschowie

mkolowsk@amu.edu.pl, marika.mackowiak@gmail.com

Abstract. *The aim of the paper is to show that certain points of the computer science curriculum at the high school level can be implemented by teaching a logic programming language. The presentation is illustrated with examples in Prolog, showing how the considered algorithms can be implemented in this logic programming language. In particular, the solution of a high school leaving exam task in Prolog is shown.*

1. Wstęp

Nauka programowania w szkole ma miejsce na wszystkich etapach edukacji. Realizacji podstawy programowej w zakresie programowania w szkole ponadpodstawowej służą języki imperatywne (Python, C++). Języki te charakteryzują się tym, że programiści wyraża w nich za pomocą rozkazów czynności, które komputer ma wykonać w określonej kolejności. Poza językami imperatywnymi istnieje grupa języków deklaratywnych, podzielona na języki funkcyjne oraz języki programowania logicznego. Pisząc program w języku deklaratywnym podaje się (deklaruje) zależności między danymi a wynikiem (celem), jaki program ma osiągnąć, nie podając wprost sposobu osiągnięcia tego wyniku [1]. Przykładami języków funkcyjnych są Haskell, Lisp, Scala, a języków programowania w logice – Prolog, Datalog, ASP.

Celem niniejszego artykułu jest pokazanie na przykładzie języka Prolog, że pewne wymagania podstawy programowej z informatyki w szkole ponadpodstawowej można zrealizować w trakcie nauki języków deklaratywnych. Na przykładzie zadania

maturalnego pokażemy, że język Prolog może być wybrany jako narzędzie do rozwiązywania zadania z baz danych na egzaminie maturalnym.

Podstawami teoretycznymi języka Prolog (**Programming in logic**) są logika pierwszego rzędu i reguła rezolucji. Program matematyki w szkole ponadpodstawowej nie obejmuje rachunku predykatów. Elementy klasycznego rachunku zdań oraz klasycznego rachunku predykatów w zakresie umożliwiającym zrozumienie podstaw programowania w Prologu są dobrym tematem do realizacji na zajęciach fakultatywnych z matematyki lub informatyki jako wstęp do programowania w języku Prolog.

2. Krótka charakterystyka języka Prolog

Prolog należy do deklaracyjnych języków programowania, zatem programu w Prologu nie zapisuje się za pomocą instrukcji charakterystycznych dla programowania imperatywnego. Program w Prologu składa się z faktów i reguł opisujących pewne relacje między obiektami [2], [3].

Zmienne i stałe. Stałe zapisuje się jako ciąg liter, cyfr i znaku podkreślenia zaczynający się małą literą lub umieszczony w apostrofach, np. `a1`, `student`, `mat_inf`, `'Szkoła'`. Nazwy zmiennych muszą zaczynać się dużą literą lub znakiem podkreślenia, np. `X`, `Y1`, `Zmienna`, `_zmienna7`. Pojedynczy znak „_” oznacza zmienną anonimową.

Termy. Termy w Prologu to zmienne, stałe i struktury. Strukturą jest wyrażenie zbudowane z funktora i argumentów będących termami, np.

`ksiazka(autor(gatnar, stapor), tytul(prolog), wydanie(pwn, warszawa, 2025)).`

Predykaty. Predykaty w Prologu opisują związki między obiektami. Nazwa predykatu musi być atomem, argumenty są dowolnymi termami. Predykaty służą do zapisywania faktów i reguł. Reguła składa się z nagłówka i treści, oddzielonych od siebie symbolem `:-`. Predykat w nagłówku jest spełniony (prawdziwy), jeżeli prawdziwe są wszystkie predykaty zapisane w treści reguły. Znak `%` jest początkiem komentarza.

Przykłady:

`owoc(jabłko) .` `% Jabłko jest owocem.`

`brat(piotr, pawel).` `% Piotr jest bratem Pawła.`

`studiuje(jan, informatyka, uam).` `% Jan studiuje informatykę na UAM.`

`siostra(X,Y) :- kobieta(X), rodzice(M,O,X), rodzice(M,O,Y), X \= Y.`

`% X jest siostrą Y, jeśli X i Y mają tych samych rodziców i X ≠ Y.`

`ojciec(X,Y) :- mezczyzna(X), rodzic(X,Y).`

`% X jest ojcem Y, jeśli jest mężczyzną i rodzicem dla Y.`

`dziadek(X,Y) :- ojciec(X,Z), rodzic(Z,Y).`

%X jest dziadkiem Y, jeśli jest ojcem rodzica Y.

Zbiór faktów tworzy bazę danych będącą zbiorem informacji, które można uzyskać zadając odpowiednie pytania.

Przykłady:

?- studiuje(X, informatyka, uam). %Kto studiuje informatykę na UAM?

?- brat(X, paweł). %Kto jest bratem Pawła?

?- dziadek(antoni,X). %Czyim dziadkiem jest Antoni?

Unifikacja. Prolog poszukując odpowiedzi (rozwiązując jakiś problem) wykorzystuje unifikację termów, która polega na poszukiwaniu unifikatora, czyli takiego podstawienia termów za zmienne, aby unifikowane termy stały się równe.

Przykłady:

stolica(warszawa,X)=stolica(Y,polska).

Unifikacja zakończy się sukcesem, a odpowiedzią Prologu jest: X=warszawa, Y=polska.

stolica(warszawa,X)=stolica(toruń,polska).

Unifikacja zakończy się porażką, gdyż stałe 'warszawa' i 'toruń' są różne.

Procedury. Procedurą jest zbiór faktów i reguł zbudowanych w oparciu o ten sam predykat.

Przykład (procedura znajdująca największy wspólny dzielnik dwóch liczb):

nwd(M, M, M).

nwd(M, N, D) :- M>N, M1 is M-N, nwd(M1, N, D).

nwd(M, N, D) :- N>M, N1 is N-M, nwd(M, N1, D).

Listy. Lista w Prologu jest strukturą rekurencyjną, składa się z głowy (pierwszy element listy) i ogona (lista złożona z pozostałych elementów).

Przykłady:

[] – lista pusta, [X|Y], lista co najmniej jednoelementowa, [a,b,c] – lista trzyelementowa.

Listy czteroelementowe unifikujące się: [1,2,3,4], [1|[2,3,4]], [1,2|[3,4]], [1,2,3|[4]].

Unifikacja [X|[Y|Z]]=[1,2,3,4] kończy się sukcesem: X=1, Y=2, Z=[3,4].

Unifikacja [X|[Y|[1]]]=[4,3,2,1] kończy się porażką z powodu różnej liczby elementów list.

Rekurencja. W Prologu nie ma instrukcji takich jak instrukcja podstawiania, instrukcja warunkowa czy pętle. Powtórzenia realizowane są z wykorzystaniem rekurencji. Procedura rekurencyjna składa się z faktu kończącego rekurencję oraz reguły lub reguł, w treści których znajduje się ten sam predykat, co w nagłówku, lecz z innymi argumentami. Przedstawiona powyżej procedura nwd jest przykładem procedury rekurencyjnej.

3. Prolog a podstawa programowa

Pierwszym celem kształcenia podstawy programowej jest „I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.” Wymaganiami szczegółowymi w tym punkcie są [6]:

„Uczeń (...):

1. planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).
2. stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:
 - 1) na liczbach: badania pierwszości liczby, zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi, działań na ułamkach z wykorzystaniem NWD i NWW,
 - 2) na tekstach: porównywania tekstów, wyszukiwania wzorca w tekście metodą naiwną, szyfrowania tekstu metodą Cezara i przestawieniową,
 - 3) porządkowania ciągu liczb: przez wstawianie i metodą bąbelkową,
 - 4) wydawania reszty najmniejszą liczbą nominalów,
 - 5) obliczania wartości elementów ciągu metodą iteracyjną i rekurencyjną, w tym wartości elementów ciągu Fibonacciego.
3. wyróżnia w problemie podproblemy i charakteryzuje: metodę połowienia, stosuje podejście zachłanne i rekurencję;
4. porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;
5. sprawdza poprawność działania algorytmów dla przykładowych danych."

Podczas rozwiązywania problemów z wykorzystaniem programowania logicznego, uczniowie mają możliwość rozwijania kompetencji wymienionych w pierwszym punkcie powyżej przedstawionych treści nauczania. Aby rozwiązać dany problem, należy bowiem rozpocząć od jego sformułowania, identyfikacji modeli i pojęć, których on dotyczy, a następnie znaleźć rozwiązanie, które da poprawne wyniki. Cały ten proces wykształci u uczniów umiejętność planowania kroków rozwiązywania problemów, nie tylko informatycznych, ale również w życiu codziennym. Ostatnimi krokami rozwiązania problemu

są napisanie programu i sprawdzenie poprawności jego działania. Poniższy przykład ilustruje rozwiązanie zadania programistycznego w języku Prolog.

Przykład 1. Polecenie: Napisz program obliczający liczbę elementów listy.

Sformułowanie problemu: Napisanie programu w języku Prolog, który policzy, ile elementów ma podawana lista i zwróci tę liczbę jako wynik działania programu.

Znalezienie rozwiązania: Napisać fakt, który stwierdza, że lista pusta ma 0 elementów. Dla listy niepustej należy napisać regułę, która mówi, że długością listy jest długość jej ogona zwiększona o 1.

Kod programu:

```
dlugosc( [ ], 0 ) .
```

```
dlugosc( [ _|Xs ] ,D) :- dlugosc(Xs , D1), D is D1 + 1 .
```

Przykład działania programu:

```
?- dlugosc([a,b,c,d,e,f],X).
```

```
    X = 6.
```

Drugi punkt wymienionych treści nauczania nawiązuje do wykorzystania algorytmów poznanych w szkole podstawowej oraz nowych, poznanych w szkole ponadpodstawowej, do rozwiązywania problemów z różnych dziedzin. Programowanie w Prologu pomoże uczniom utrwalić wcześniej zdobytą wiedzę oraz wykształci u nich umiejętność implementowania algorytmów w nowym języku programowania.

Zaprezentujemy teraz wybrane algorytmy z punktu 2. podstawy programowej zapisane w języku Prolog.

Przykład 2. Sortowanie elementów ciągu liczbowego metodą bąbelkową.

```
bubble_sort( List, Sorted ):- b_sort( List, [ ], Sorted).
```

```
b_sort( [ ], Acc, Acc ).
```

```
b_sort( [ H|T ], Acc, Sorted):-bubble( H, T, NT, Max) ,  
                                b_sort(NT, [Max| Acc ], Sorted).
```

```
bubble(X, [ ], [ ], X).
```

```
bubble(X, [Y|T], [Y|NT], Max):-X>Y, bubble(X, T, NT, Max).
```

```
bubble(X, [Y|T], [X|NT], Max):-X<=Y, bubble(Y, T, NT, Max).
```

Przykład działania procedury:

```
?- bubble_sort([7,6,4,5,2,3,1], X).
```

```
X = [1,2,3,4,5,6,7].
?- bubble_sort(5,1,9,2,5,3,2],Y).
Y = [1,2,2,3,5,5,9].
```

Przykład 3. Badanie pierwszości liczb.

```
czy_pierwsza(2).
czy_pierwsza(3).
czy_pierwsza(X) :- X > 3, X mod 2 =\= 0, czy_pierwsza2(X, 3).
czy_pierwsza2(X,N) :- N*N > X; (X mod N =\= 0, M is N + 2,
                           czy_pierwsza2(X,M)).
```

Przykład działania procedury:

```
?-czy_pierwsza(11).
true.
?- czy_pierwsza(100).
false.
```

Przykład 4. Obliczanie n-tego wyrazu ciągu Fibonacciego.

```
fib(1,1).
fib(2,1).
fib(X,Y) :- X>2, X1 is X-1, X2 is X-2, fib(X1 ,Y1), fib(X2 ,Y2), Y is Y1+Y2 .
```

Przykład działania procedury:

```
?- fib(5, X).
X = 5.
?- fib(8, X).
X = 21.
```

Przykład 5. Wydawanie reszty najmniejszą liczbą nominałów (w złotych).

```
moneta(500). moneta(200). moneta(100).
moneta(50). moneta(20). moneta(10).
moneta(5). moneta(2). moneta(1).
reszta(0, [ ]).
reszta(X, [ [Y | Ys] ] ) :- X>0, moneta (Z), Z<X, A is div(X, Z), B is mod(X, Z),
                           Y = [Z ,A], reszta(B, Ys).
```

Przykład działania procedury:

?-reszta(641,X).

X = [[500, 1], [100, 1], [20, 2], [1, 1]].

?-reszta(1649,X).

X = [[500, 3], [100, 1], [20, 2], [5, 1], [2, 2]].

Przykład 6. Zaszzyfrowanie komunikatu zapisanego wielkimi literami alfabetu łacińskiego szyfrem Cezara.

szyfr_Cezara([], []).

szyfr_Cezara(X,Y):- string_to_list(X,X1), cezar(X1 ,Y1), cezar2(Y1, Y2),
atomics_to_string(Y2, Y).

cezar([], []).

cezar([H|T], [H1|Y1]) :- H + 3 > 90, H1 is H+3-26, cezar(T,Y1).

cezar([H|T],[H1 |Y1]) :- H + 3 < 65, H1 is H+3+26, cezar(T,Y1).

cezar([H|T], [H1|Y1]) :- H1 is H+3, cezar (T,Y1).

cezar2([], []).

cezar2([H1|T1], [H2|T2]) :- char_code(H2, H1), cezar2(T1, T2).

Przykład działania procedury:

?- szyfr_Cezara("ABC",X).

X = "DEF".

?- szyfr_Cezara("XYZ",X).

X = "ABC".

4. Rozwiązanie zadania maturalnego w Prologu

W tym rozdziale pokażemy, że zadanie maturalne związane z wyszukiwaniem informacji w bazie danych można rozwiązać wykorzystując język programowania Prolog. Zadanie pochodzi z matury w 2023 roku [5]:

Zadanie 5. Księgarnia internetowa. Księgarnia internetowa „InterBook” posiada szeroką ofertę e-booków i audiobooków z różnych dziedzin nauki. W plikach publikacje.pl i dziedziny.pl zawarte są informacje na ich temat. Natomiast plik zakupy.pl zawiera informacje o transakcjach zakupu tych publikacji w ciągu jednego miesiąca. Pierwszy wiersz każdego z plików jest wierszem nagłówkowym, a szczegółowe informacje w każdym wierszu rozdzielone są znakami tabulacji. Plik o nazwie publikacje.pl zawiera 2000 wierszy

z informacjami na temat oferowanych publikacji. W każdym wierszu znajdują się: identyfikator publikacji, autor, tytuł, wydawnictwo, rok wydania, format pliku, cena oraz identyfikator dziedziny nauki. Przykład:

```
%ID_pub.,Autor,Tytuł,Wydawnictwo,Rok_wyd.,Format,Cena,ID_dziedziny
```

```
publikacja(e1386,'Miroslawa Kopertowska, Witold Sikorski', 'Funkcje w Excelu. ',  
'Wydawnictwo Naukowe PWN ',2009,epub,4.99,d02).
```

```
publikacja(a2069,'James M. Barrie ', 'Piotrus Pan i Wendy ', 'Biblioteka Akustyczna  
'2015,mp3,24.99,d04).
```

W pliku dziedziny.pl zapisano 7 wierszy zawierających identyfikatory dziedzin nauki oraz ich nazwy. Przykład:

```
%id_dziedziny,dziedzina
```

```
dziedzina(d01,ekonomia).
```

```
dziedzina(d02,informatyka).
```

Plik zakupy.pl zawiera 2500 wierszy opisujących transakcje zakupu publikacji przez klientów w marcu 2017 roku. Przykład:

```
%nick_klienta, data_zakupu, zid_publicacji
```

```
zakup(witu,'01.03.2017',e1669).
```

```
zakup(huanpablo,'01.03.2017',a1284).
```

Wykorzystując dane zapisane w plikach oraz dostępne narzędzia informatyczne, wykonaj podane zadania. Odpowiedzi zapisz w pliku wyniki5.txt, a każdą odpowiedź poprzedź numerem odpowiedniego zadania.

Zadanie 5.1. (2 pkt) Identyfikatory audiobooków zaczynają się literą a, natomiast e-booków – literą e. Podaj tytuły wszystkich audiobooków droższych niż 45 zł i wszystkich e-booków w cenie powyżej 120 zł. Posortuj te publikacje malejąco według cen.

Zadanie 5.2. (2 pkt) Podaj autorów i tytuły publikacji z informatyki lub ekonomii wydanych w 2014 roku, zakupionych przez klienta o nicku *flipper*.

Zadanie 5.3. (2 pkt) Dla każdej dziedziny nauki podaj średnią cenę publikacji (w formacie walutowym) z tej dziedziny.

Zadanie 5.4. (2 pkt) Podaj nazwę dziedziny, z której kupiono najwięcej publikacji.

Zadanie 5.5. (2 pkt) Podaj nick klienta, który wydał największą kwotę na wszystkie zakupione publikacje, oraz nick klienta, który wydał najmniej. Dla każdego z nich podaj też odpowiednią kwotę.

Wszystkie rozwiązania zapisane są w jednym pliku, który rozpoczyna się wczytaniem bazy danych podanej w zadaniu do pamięci:

`:- [dziedziny, publikacje, zakupy].`

Zadanie 5.1. Przykładowy kod programu:

```
lista_e(X) :- findall([A,T,C],(publikacja(A,_,T,_,_,C, _),
    string_to_list(A,[L|_]),L=101, C>120),X).
lista_a(X) :- findall([A,T,C], (publikacja(A,T,_,_,_,C, _),
    string_to_list(A,[L|_]), L = 97, C>45),X).

bubble_sort(List, Sorted) :- b_sort(List, [], Sorted).
b_sort([], Acc, Acc).
b_sort([H|T], Acc, Sorted) :- bubble(H, T, NT, Max),
    b_sort(NT,[Max|Acc],Sorted).

bubble(X,[],[],X).
bubble([A,B,X],[[C,D,Y]|T],[[C,D,Y]|NT],Max):-X<=Y,bubble([C,D,Y],T,NT,Max).
bubble([A,B,X],[[C,D,Y]|T],[[A,B,X]|NT],Max):-X>Y, bubble([C,D,Y],T,NT,Max).
zadanie1 :- lista_e(A),lista_a(B), append(A,B,X1),
    bubble_sort(X1,X),maplist(writeln,X).
```

Do wykonania zadania zostały wykorzystane trzy procedury: `lista_e`, `lista_a`, `zadanie1` oraz procedura `bubble_sort`. Procedury: `lista_e` i `lista_a` działają podobnie, z tą różnicą, że pierwsza z nich wyszukuje e-booki, a druga audiobooki. Obie procedury do znalezienia określonych w treści zadania publikacji wykorzystują predykat wbudowany `findall`. Pierwszym argumentem przyjmowanym przez ten predykat, jest lista `[A,T,C]`, czyli element, którego szukamy w bazie danych. Drugim argumentem są warunki poszukiwania elementu, o którym mowa w zadaniu. Tymi warunkami są: spośród publikacji w bazie danych zwróć uwagę na indeks publikacji `A`, tytuł publikacji `T` oraz cenę publikacji `C`, gdzie pierwsza litera indeksu publikacji ma kod ASCII równy 101 (odpowiada literze 'e'), (w wersji dla e-booków mamy 97, co odpowiada literze 'a'), a cena jest większa odpowiednio od 120 lub 45. Procedury te zwracają wszystkie publikacje spełniające warunki z drugiego argumentu jako listę `X`. Procedura `zadanie1` jest bezargumentowa ponieważ wypisuje jedynie listy jako wynik działania opisanych procedur. Ze względu na zachowanie cen publikacji w kolejności malejącej, została zastosowana również procedura będąca implementacją algorytmu sortowania bąbelkowego.

Następnie dzięki użyciu procedury `maplist`, która nakłada na każdy element listy `X` procedurę `writeln`, otrzymujemy jako wynik wypisanie wymaganych w treści zadania publikacji w odpowiedniej kolejności. Wbudowana procedura `writeln` powoduje wypisanie na ekranie termu podanego jako argument.

Zadanie 5.2. Przykładowy kod programu:

```
zadanie2(X,Y,Z) :- zakup(flipher,_,I), publikacja(I,X,Y,_,2014,_,_,D),
                    dziedzina(D,Z), (Z=ekonomia ; Z=informatyka).
```

Procedura `zadanie2` działa następująco: 1. Przeszukuje bazę danych z zakupami wykonanymi w sklepie i poszukuje tych faktów, gdzie `nick` kupującego jest równy `flipher`, po unifikacji zakończonej sukcesem, ukonkretnia zmienną `I` z numerem indeksu publikacji. 2. Przeszukuje bazę danych z publikacjami i szuka tych danych, dla których indeks jest równy znalezionej wartości `I`, a rok wydania jest równy 2014 oraz zapisuje autora `X`, tytuł `Y`, identyfikator dziedziny `D` znalezionych faktów. 3. Przeszukuje bazę danych z kategoriami publikacji i zapisuje nazwy znalezionych w punkcie 2 dziedzin jako `Z`. 4. Sprawdza, czy nazwą dziedziny `Z` jest 'ekonomia' lub 'informatyka'.

Jeśli wszystkie predykaty będą spełnione, to Prolog zwróci autora, tytuł i dziedzinę publikacji. Aby uzyskać wszystkie tytuły należy) zamiast klawisza ENTER wcisnąć średnik (;) (wymuszający w Prologu alternatywne odpowiedzi).

Zadanie 5.3. Przykładowy kod programu:

```
round(X,Y,D) :-- Z is X * 10^D, round(Z, ZA), Y is ZA/10^D.
f1(D,S) :- aggregate_all(sum(X),publikacja(_,_,_,_,_,X,D),A),
           aggregate_all(count,publikacja(_,_,_,_,_,X,D), B), S1 is A/B,
           round(S1,S,2).

zadanie3([B,S]) :- dziedzina(A,B) , f1(A,S).
```

Do znalezienia odpowiedzi do zadania trzeciego, zostały napisane trzy procedury: `round`, `f1` i `zadanie3`. Procedura `round` służy do zaokrąglania liczby `X` do liczby `D` miejsc po przecinku i zwraca wynik jako `Y`.

Procedura `f1` wykorzystuje procedurę `aggregate_all` do zsumowania cen wszystkich publikacji, których numer indeksu dziedziny jest równy `D` oraz do policzenia, ile jest łącznie publikacji w tej dziedzinie. Następnie oblicza średnią `S1` i zaokrągla ją do dwóch miejsc po przecinku.

Procedura `zadanie3` dla każdej dziedziny z bazy z dziedzinami oblicza średnią cen za publikacje oraz tworzy listę wynikową składającą się z nazwy dziedziny oraz średniej cen za publikacje.

Zadanie 5.4. Przykładowy kod programu:

```
ile(D,L) :- aggregate_all(count,(zakup(__,__X), publikacja(X,__,__D)),L).
f2([B,L]) :- dziedzina(A, B), ile(A,L).
f3(X) :- findall(A, f2(A), X).
list_max([P|T], O) :- list_max(T, P, O).
list_max([], [K,P], [K,P]).
list_max([[K,H]|T], [L,P], O) :- (H > P -> list_max(T, [K,H], O);
                                list_max(T, [L,P], O)).

zadanie4(X) :- f3(A), list_max(A,X).
```

Do wykonania zadania czwartego zostało użytych pięć procedur:

ile - wykorzystuje procedurę `aggregate_all` do znalezienia liczby dokonanych zakupów L publikacji z kategorii D.

f2 - szuka dziedzin publikacji i oblicza dla nich liczbę wszystkich dokonanych zakupów wykorzystując poprzednią procedurę, następnie tworzy listę wynikową składającą się z nazwy dziedziny i liczby dokonanych zakupów publikacji z tej kategorii.

f3 - tworzy listę składającą się ze wszystkich znalezionych przez procedurę f2 list.

list_max - poszukuje na liście złożonej z list takiej listy, której drugi element jest największy spośród drugich elementów wszystkich list.

zadanie4 - wywołuje procedurę f3 do znalezienia listy A składającej się z dziedzin i liczby zakupów publikacji z tej dziedziny, oraz wywołuje procedurę list_max do znalezienia nazwy dziedziny, z której publikacje zostały najwięcej razy kupione.

Zadanie 5.5. Przykładowy kod programu:

```
ile1(N,K) :- aggregate_all(sum(X), (zakup(N,__,A),
                                   publikacja(A,__,__X,_)), K1), round(K1, K, 2).

f4([A,K]) :- zakup(A,__,_), ile1(A, K).

f5(X) :- findall(A, f4(A), X).
list_min([], [K,P], [K,P]).
list_min([P|T], O) :- list_min(T, P, O).
list_min([[K,H]|T], [L,P], O) :- (H < P -> list_min(T, [K,H], O);
                                list_min(T, [L,P], O)).

zadanie5 :- f5(A), list_min(A,X1), list_max(A,X2), maplist(writeln,[X2,X1]).
```

Wykorzystane procedury:

ile1 - oblicza sumę cen X publikacji A zakupionych przez użytkownika o nicku N i zaokrągla wynik do dwóch miejsc po przecinku (wykorzystanie procedury z zadania 5.3.).

f4 - dla każdego użytkownika A , który dokonał zakupów, sprawdza ile wynosiła łączna kwota wydana na publikacje K i zwraca wynik X jako listę nicków i kwot.

f5 - tworzy listę X z wyników otrzymanych przez wywołanie procedury f4.

list_min - poszukuje na liście takiej listy, której drugi element jest najmniejszy spośród drugich elementów wszystkich list.

Zadanie5 - tworzy listę A zawierającą wydane kwoty na publikację przez każdego użytkownika, szuka minimalnej $X1$ i maksymalnej $X2$ wydanej kwoty i przy użyciu procedur maplist i writeln wypisuje użytkownika, który wydał najwięcej i najmniej pieniędzy łącznie z kwotami.

Uwaga. Wszystkie procedury zwracają wyniki, które są poprawnymi odpowiedziami, zgodnymi z kluczem odpowiedzi do zadań maturalnych z informatyki.

5. Podsumowanie

Zajęcia fakultatywne z informatyki w szkole ponadpodstawowej mogą być poświęcone językom programowania innym niż te, które uczniowie poznają na zajęciach (Python, C++), mogą to być języki programowania deklaratywnego. Jak pokazano powyżej, umiejętność programowania w Prologu może być wykorzystana na maturze z informatyki. Podobnie, pewne zadania maturalne można rozwiązać używając języków programowania funkcyjnego, co autorki pokazały w pracy [4].

Nauka programowania w Prologu w szkole ponadpodstawowej uczy logicznego i twórczego myślenia, wskazuje na związki matematyki z informatyką, ilustrując w szczególności zastosowania logiki w informatyce.

Literatura

1. Bylina J., Bylina B., Przegląd języków i paradygmatów programowania, Uniwersytet Marii Curie-Skłodowskiej w Lublinie, Instytut Informatyki, Lublin 2011.
2. Clocksin W. F., Mellish C. S., Prolog Programowanie, Wydawnictwo Helion, Gliwice 2003.
3. Gatnar E., Stapor K., Prolog. Język Sztucznej Inteligencji, Wydawnictwo PLJ, Warszawa 1991.

4. Kołowska-Gawiejnowicz M., Maćkowiak M., Programowanie funkcyjne w szkole ponadpodstawowej, Informatyka w edukacji, red. Kwiatkowska Anna Beata, Sysło Maciej M., Wydawnictwo Adam Marszałek (2023), str. 148-158.
5. Arkusz maturalny, matura z informatyki na poziomie podstawowym,
6. <https://arkusze.pl/maturalne/informatyka-2019-maj-matura-stara-podstawowa-2.pdf> (dostęp:02.06.2025)
7. Rozporządzenie Ministra Edukacji z dnia 28 czerwca 2024 r. zmieniające rozporządzenie w sprawie podstawy programowej kształcenia ogólnego dla liceum ogólnokształcącego, technikum oraz branżowej szkoły II stopnia, <https://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20240001019> (dostęp:02.06.2025)