INSTITUT ZA MATEMATIKU I INFORMATIKU

PRIRODNO-MATEMATIČKI FAKULTET

SEMINARSKI RAD

Predmet: Logičko i funkcijsko programiranje

Tema: Moj Broj – Igrica iz famoznog kviza Slagalica

Student:
Stefan Aleksandrić
43/2016

Mentor:
Tatjana Stojanović

Sadržaj

1.Opis i ideja igre

Igra **"Moj Broj"** je jedan od najpoznatijih i najpopularnijih segmenata kviza **"Slagalica"**, koji se emituje na Radio-televiziji Srbije. "Slagalica" je jedan od najdugovečnijih i najgledanijih televizijskih kvizova u Srbiji, a kroz svoje različite igre stavlja na probu opšte znanje, logičko razmišljanje, i matematičke sposobnosti takmičara.

U igri "Moj Broj", takmičari dobijaju zadatak da pomoću šest nasumično generisanih brojeva i osnovnih matematičkih operacija (+, -, *, /) dođu do ciljanog broja koji je takođe nasumično generisan, i nalazi se između 100 i 999.

Pravila igre:

1. **Generisani brojevi:**
   Takmičari biraju šest brojeva iz dva skupa – 4 mala broja (od 1 do 10) i 2 velika broja (prvi od 10 do 49, drugi od 50 do 100). Oni zatim koriste ove brojeve kako bi izračunali zadati ciljani broj.

2. **Cilj igre:**
   Cilj je da se, koristeći dostupne brojeve i osnovne matematičke operacije, dođe do ciljanog broja ili što bliže njemu i na taj način osvojiti poene.

3. **Poeni:**
   Takmičari koji tačno reše zadatak dobijaju maksimalan broj poena (10 poena), dok takmičari koji su blizu cilju dobijaju poene na osnovu toga koliko su blizu traženom broju. U našem slučaju za približni broj to će biti 5 poena ukoliko rešenje takmičara bude bliže ciljanom broju od rešenja drugog takmičara.

Zanimljivosti o igri "Moj Broj" i kvizu "Slagalica"

"Slagalica" je simbol inteligentnog zabavnog programa i zahteva različite veštine od učesnika – od znanja reči (kroz igru "Spojnice" i "Asocijacije"), do logike i matematike, što se najbolje vidi u igri "Moj Broj". Publika je kroz godine zavolela ovaj segment jer kombinuje brzinu razmišljanja sa sposobnošću da se brzo koriste brojevi i operacije.

Zašto je "Moj Broj" interesantan?

"Moj Broj" posebno ističe takmičarsku atmosferu jer poziva takmičare da na inovativan i brz način koriste osnovne matematičke operacije. Upravo ova igra pruža mogućnost publici kod kuće da se i oni igraju i pokušaju da "nađu broj" pre takmičara, što je dodatno čini omiljenom među gledateljima.

Implementacija igre

U ovoj implementaciji igre koristiće se logički programski jezik **Prolog** i takmičari će jedan za drugim unositi svoje najbolje rešenje. Prvo prvi takmičar pa zatim drugi. Igra će se vrteti u krug sve dok se ne unese komanda za zaustavljanje igre nakon čega će se takmičaru sa većim brojema poena proglasiti pobeda.

2.Uputstvo za korišćenje

Za pokretanje igre potrebno je instalirati programsko okruženje za rad SWI – Prolog koje se može preuzeti sa sledeće web stranice za Windows operativni sistem:
- https://www.swi-prolog.org/download/stable

Dok za Linux operativne sisteme možemo instalirati podršku za Prolog pomoću komande:

- **sudo apt-get install swi-prolog**

Igra se pokreće učitavanjem Prolog programskog okruženja i pokretanjem komandi **[mojBroj].** i ukoliko je sve u redu pokretanjem komande **play_game.** :

```
ritamnereda@SamuraiJack:~/LiF/prolog/Slagalica$ prolog
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [mojBroj].
true.

?- play_game.
Dobrodošli u igru Moj Broj!

Ciljni broj: 667
Dostupni brojevi: [5,1,3,5,49,83]
```

3. Kod i objašnjenje

```
1   :- use_module(library(random)). % Za generisanje random brojeva u opsezima koji se koriste
2
```

Ova linija koda :- use_module(library(random)). uvozi Prologovu biblioteku za generisanje nasumičnih brojeva. To omogućava igri da kreira nasumične brojeve unutar definisanih opsega, što je ključno za dinamičnost i nepredvidivost tokom igranja.

```
3   % Dinamički predikat za praćenje poena igrača
4   :- dynamic scores/2.
5   scores(0, 0).  % Početni poeni igrača 1 i igrača 2
```

Ovaj deo koda definiše dinamički predikat scores/2, koji se koristi za praćenje poena dva igrača. Dinamički predikati omogućavaju promenu svojih vrednosti tokom izvođenja programa.

Linija scores(0, 0). postavlja početne poene za igrača 1 i igrača 2 na nulu, što znači da igra počinje bez poena. Ova struktura omogućava igri da ažurira i prati rezultate tokom svakog kruga.

```
% Generisanje nasumičnog broja između Min i Max
random_number(Min, Max, Num) :-
    random_between(Min, Max, Num).

% Generisanje šest manjih brojeva
generate_numbers(Numbers) :-
    findall(X, (between(1, 4, _), random_number(1, 9, X)), SingleDigitNumbers),
    random_number(10, 49, BiggerNumber1),
    random_number(50, 100, BiggerNumber2),
    append(SingleDigitNumbers, [BiggerNumber1, BiggerNumber2], Numbers).

% Generisanje ciljnog broja i šest brojeva koji se koriste za igru
start_game(Target, Numbers) :-
    random_number(100, 999, Target),
    generate_numbers(Numbers).
```

Ovaj deo koda se koristi za generisanje nasumičnih brojeva koji će se koristiti u igri.

1. **random_number/3**: Ova funkcija generiše nasumičan broj između Min i Max, a rezultat se dodeljuje varijabli Num. Funkcija koristi biblioteku random koja pruža mogućnosti za generisanje slučajnih brojeva.
2. **generate_numbers/1**: Ova funkcija generiše listu brojeva koja sadrži šest manjih brojeva. Prvo, koristi findall/3 da generiše četiri nasumična broja u opsegu od 1 do 9 (pojedinačni cifri). Zatim generiše jedan veći broj u opsegu od 10 do 49 i još jedan veći broj u opsegu od 50 do 100. Na kraju, svi generisani brojevi se kombinuju u listu Numbers.
3. **start_game/2**: Ova funkcija pokreće igru generišući jedan ciljni broj u opsegu od 100 do 999, koristeći random_number/3, kao i šest brojeva pozivanjem generate_numbers/1. Rezultati se vraćaju kroz varijable Target (ciljni broj) i Numbers (listu generisanih brojeva).

Ove funkcije čine osnovu za mehaniku igre, omogućavajući nasumičnu generaciju potrebnih brojeva pre početka igre.

```
% Proverava da li je izraz dobar
check_expression(Expression, Result) :-
    catch(
        ( Result is Expression,
          ( integer(Result) -> true ; fail)  % Proverava da li je rezultat celi broj
        ),
        _,
        fail
    ).

% Proverava validnost izraza u odnosu na dostupne brojeve
valid_expression(Expression, AvailableNumbers) :-
    extract_numbers(Expression, UsedNumbers),
    valid_number_usage(UsedNumbers, AvailableNumbers).
```

Ovaj deo koda sadrži dve funkcije koje se koriste za proveru ispravnosti izraza u igri.

1. **check_expression/2**:
    - Ova funkcija proverava da li je dati izraz (Expression) validan i da li rezultira celim brojem (Result).
    - Koristi catch/3 kako bi uhvatila sve greške koje mogu nastati prilikom evaluacije izraza.
    - Ako se izraz uspešno evaluira, proverava se da li je rezultat celobrojni (koristeći integer/1). Ako jeste, funkcija se smatra uspešnom. Ako nije, korišćenje fail označava neuspeh. U slučaju da dođe do greške prilikom evaluacije izraza, takođe se vraća neuspeh.
2. **valid_expression/2**:
    - Ova funkcija proverava da li izraz koristi samo dostupne brojeve.
    - Koristi pomoćnu funkciju extract_numbers/2 da bi izvukla brojeve koji se koriste u izrazu (UsedNumbers).
    - Zatim poziva valid_number_usage/2 da proveri da li su svi korišćeni brojevi u UsedNumbers prisutni u listi AvailableNumbers.

Ove funkcije su ključne za validaciju izraza koji igrači mogu sastaviti u igri, osiguravajući da su izrazi tačni i da koriste samo dostupne brojeve.

```
% Broji koliko puta se element pojavljuje u listi
count(_, [], 0).
count(X, [X | Tail], Count) :-
    count(X, Tail, TempCount),
    Count is TempCount + 1.
count(X, [_ | Tail], Count) :-
    count(X, Tail, Count).

% Ekstraktovanje svih brojeva iz izraza u listu
extract_numbers(Expression, Numbers) :-
    (   number(Expression) ->
        Numbers = [Expression];
        Expression =.. [_Op | Args],
        extract_numbers_from_list(Args, Numbers)
    ).

% Ekstraktovanje brojeve iz liste argumenata
extract_numbers_from_list([], []).
extract_numbers_from_list([Arg | Rest], Numbers) :-
    extract_numbers(Arg, NumbersArg),
    extract_numbers_from_list(Rest, NumbersRest),
    append(NumbersArg, NumbersRest, Numbers).
```

Ovaj deo koda se bavi ekstrakcijom brojeva iz izraza i brojanjem ponavljanja elemenata u listi.

1. **count/3**:
    - Ova funkcija broji koliko puta se određeni element (X) pojavljuje u listi.
    - **Osnovni slučaj**: Ako je lista prazna ([]), broj ponavljanja je 0.
    - **Rekurzivni slučaj**: Ako je prvi element liste ([X | Tail]) isti kao X, funkcija se rekurzivno poziva na rep liste (Tail) i povećava broj ponavljanja za 1.
    - Ako prvi element nije jednak X, funkcija se ponovo poziva na rep liste bez povećanja broja.
2. **extract_numbers/2**:
    - Ova funkcija ekstraktuje sve brojeve iz izraza (Expression) i stavlja ih u listu (Numbers).
    - Ako je izraz broj (number(Expression)), dodaje ga u listu.
    - Ako nije broj, koristi operator =.. za razlaganje izraza na operator i argumente, a zatim poziva extract_numbers_from_list/2 za ekstrakciju brojeva iz liste argumenata.

3. **extract_numbers_from_list/2**:
    - Ova pomoćna funkcija ekstraktuje brojeve iz liste argumenata.
    - **Osnovni slučaj**: Ako je lista prazna, vraća praznu listu.
    - **Rekurzivni slučaj**: Ekstraktuje brojeve iz prvog argumenta (Arg) i ostatka liste (Rest), a zatim spaja rezultante koristeći append/3.

Ovaj deo koda je ključan za analizu izraza koje igrači mogu sastaviti, omogućavajući igri da proveri koje brojeve igrači koriste i koliko puta su ti brojevi korišćeni.

```
% Proverava da li su brojevi korišćeni validno u odnosu na dostupne brojeve
valid_number_usage([], _).
valid_number_usage([Number | Rest], AvailableNumbers) :-
    count(Number, AvailableNumbers, AvailableCount),   % Koliko puta je broj dostupan
    count(Number, [Number | Rest], UsedCount),         % Koliko puta je broj korišćen
    UsedCount =< AvailableCount,                       % Korišćenje broja ne sme biti više od dostupnog
    delete_one(AvailableNumbers, Number, NewAvailable),  % Ukloni jedan broj iz liste dostupnih brojeva
    valid_number_usage(Rest, NewAvailable).            % Proveri ostatak

% Uklanja jedno pojavljivanje broja iz liste
delete_one([X | Tail], X, Tail).
delete_one([Y | Tail], X, [Y | NewTail]) :-
    delete_one(Tail, X, NewTail).
```

Ovaj deo koda se fokusira na validaciju korišćenja brojeva u izrazu u odnosu na dostupne brojeve.

1. **valid_args/2**:
    - Ova funkcija proverava da li su svi argumenti (brojevi i operacije) u izrazu validni i da li su dostupni.
    - **Osnovni slučaj**: Prazna lista ([]) je uvek validna.
    - **Rekurzivni slučaj**: Ako je argument (Arg) broj, koristi member/2 da proveri da li je taj broj prisutan u listi dostupnih brojeva (Numbers).
    - Ako Arg nije broj, pretpostavlja se da je operator (koristeći =..), pa se proverava da li je taj operator jedan od validnih (+, -, *, /) i rekurzivno proveravaju njegovi argumenti.
2. **valid_number_usage/2**:
    - Ova funkcija proverava da li su brojevi korišćeni validno u izrazu, u odnosu na dostupne brojeve.
    - **Osnovni slučaj**: Prazna lista ([]) znači da su svi brojevi validni.
    - **Rekurzivni slučaj**: Za svaki broj u listi:
        - count/3 se koristi da izbroji koliko puta je broj (Number) dostupan (AvailableCount) i koliko puta je korišćen (UsedCount).

- Proverava se da korišćenje broja ne prelazi broj dostupnih instanci (UsedCount =< AvailableCount).
- Ako je broj validan, delete_one/3 se koristi da ukloni jedan primerak tog broja iz liste dostupnih brojeva, a zatim se rekurzivno proverava ostatak liste.

3. **delete_one/3**:
    - Ova pomoćna funkcija uklanja jedno pojavljivanje broja iz liste.
    - **Osnovni slučaj**: Ako je prvi element liste (X) isti kao broj koji treba ukloniti, vraća ostatak liste (Tail).
    - **Rekurzivni slučaj**: Ako prvi element nije isti, funkcija se poziva rekurzivno na rep liste da potraži broj koji treba ukloniti.

Ovaj deo koda je bitan za osiguravanje da igrači koriste dostupne brojeve na ispravan način, čime se povećava fer igru i izazov.

```
86  % Logika za početak igre
87  play_game :-
88      write('Dobrodošli u igru Moj Broj!'),nl,
89      start_game(Target, Numbers),nl,
90      write('Ciljni broj: '), write(Target), nl,
91      write('Dostupni brojevi: '), write(Numbers), nl,nl,
92      play_round(Target, Numbers).
93
```

Ovaj deo koda predstavlja logiku za pokretanje igre "Moj Broj".

1. **play_game/0**:
    - **Započinje igru**: Poziva se kada igrač želi da igra "Moj Broj".
    - **Dobrodošlica**: Prvo ispisuje poruku dobrodošlice, "Dobrodošli u igru Moj Broj!".
    - **Pokretanje igre**: Poziva start_game/2, koja generiše ciljni broj (Target) i listu dostupnih brojeva (Numbers). Ovi brojevi se koriste u igri.
    - **Ispis ciljnog broja**: Ispisuje ciljni broj koji igrači treba da postignu, zajedno sa dostupnim brojevima koje mogu koristiti.
    - **Poziv runde**: Poziva play_round/2 sa generisanim ciljnim brojem i dostupnim brojevima kako bi započela igra.

Ovaj deo koda je ključan za inicijalizaciju igre i omogućava igračima da vide ciljni broj i brojeve koje mogu koristiti za postizanje tog broja. On postavlja scenu za igru i pokreće interakciju sa igračem.

```
% Logika za jednu rundu
play_round(Target, Numbers) :-
    % Igrač 1
    write('Igrač 1 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!): '), nl,
    read(Expression1),
    handle_player_input(1, Expression1, Target, Numbers, Result1),
    % Igrač 2
    write('Igrač 2 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!): '), nl,
    read(Expression2),
    handle_player_input(2, Expression2, Target, Numbers, Result2),
    % Provera pobednika
    check_winner(Result1, Result2, Target),
    % Ispis trenutnih poena
    print_scores,
    % Pitanje za nastavak igre
    continue_game.
```

Ovaj deo koda implementira logiku za jednu rundu igre "Moj Broj", omogućavajući interakciju između dva igrača.

1. **play_round/2**:
    - Prihvaća Target (ciljni broj) i Numbers (dostupni brojevi) kao ulazne argumente.
2. **Igrač 1**:
    - Ispisuje poruku koja traži od Igrača 1 da unese izraz koristeći dostupne brojeve i osnovne matematičke operacije.
    - Čita uneti izraz iz ulaza i skladišti ga u promenljivu Expression1.
    - Poziva handle_player_input/5, koja obrađuje unos igrača, proverava validnost izraza i izračunava rezultat. Rezultat se skladišti u Result1.
3. **Igrač 2**:
    - Ponavlja istu proceduru kao za Igrača 1, tražeći od Igrača 2 da unese svoj izraz. Rezultat se skladišti u Result2.
4. **Provera pobednika**:
    - Poziva check_winner/3, koja proverava ko je bliži ciljnog broju na osnovu rezultata oba igrača.
5. **Ispis trenutnih poena**:
    - Poziva print_scores/0 da ispiše trenutne poene za oba igrača nakon runde.
6. **Nastavak igre**:
    - Poziva continue_game/0 kako bi igra pitala igrače da li žele da nastave ili završe igru.

```
% Obrada unosa igrača
handle_player_input(Player, Expression, Target, Numbers, Result) :-
    (   valid_expression(Expression, Numbers),   % Proveri validnost izraza
        check_expression(Expression, Result) ->  % Proveri rezultat izraza
        nl, write('Igrač '), write(Player), write(' je dobio broj: '), write(Result), nl, nl;

        % Ako je nevalidan izraz
        nl, write('Uneseni izraz igrača '), write(Player), write(' nije validan ili su brojevi korišćeni više puta nego što je dostupno.'), nl, nl,
        Result = 0  % Igrač ne dobija poene
    ),
    _ = Target.

```

Ovaj deo koda upravlja obradom unosa igrača u igri "Moj Broj" i sadrži logiku za proveru validnosti izraza koji igrači unose.

1. **handle_player_input/5**:
    - Prihvaća sledeće argumente: Player (broj igrača), Expression (uneseni izraz), Target (ciljni broj), Numbers (dostupni brojevi) i Result (rezultat izraza).
2. **Provera validnosti izraza**:
    - Poziva valid_expression/2 da proveri da li je uneti izraz validan u odnosu na dostupne brojeve.
    - Ako je izraz validan, koristi check_expression/2 za izračunavanje rezultata izraza.
    - Ako su oba uslova ispunjena:
        - Ispisuje poruku koja obaveštava o broju koji je igrač dobio i skladišti rezultat u Result.
    - U suprotnom (ako je izraz nevalidan):
        - Ispisuje poruku o grešci koja ukazuje da uneti izraz nije validan ili da su brojevi korišćeni više puta nego što je dostupno.
        - Skladišti 0 kao rezultat, što znači da igrač ne dobija poene.
3. **_ = Target**:
    - Ovaj deo koda nije funkcionalan jer _ predstavlja neodređenu varijablu. Ovaj deo se može ukloniti ili promeniti kako bi imao smisla u kontekstu logike igre. Stavljen je samo kako bi se sakrilo upozorenje.

```
% Proverava pobednika na osnovu brojeva
check_winner(Result1, Result2, Target) :-
    (   Result1 =:= Target, Result2 =:= Target ->
        add_score(1, 5),  % Igrač 1 dobija 5 poena
        add_score(2, 5),  % Igrač 2 dobija 5 poena
        nl,write('Oba igrača su pogodila tačan broj!'), nl;
        Result1 =:= Target ->
        add_score(1, 10),  % Igrač 1 dobija 10 poena
        nl,write('Igrač 1 je pogodio tačan broj!'), nl;
        Result2 =:= Target ->
        add_score(2, 10),  % Igrač 2 dobija 10 poena
        nl,write('Igrač 2 je pogodio tačan broj!'), nl;
        Result1 = 0, Result2 = 0 ->
        nl,write('Oba igrača su uneli nepravilne izraze.'), nl;
        Result1 =\= 0, Result2 =\= 0 ->
        Diff1 is abs(Result1 - Target),
        Diff2 is abs(Result2 - Target),
        (   Diff1 < Diff2 ->
            add_score(1, 5),  % Igrač 1 dobija 5 poena
            nl,write('Igrač 1 je pobedio rundu jer je uneo približniji broj ciljanom broju!'), nl;
            Diff2 < Diff1 ->
            add_score(2, 5),  % Igrač 2 dobija 5 poena
            nl,write('Igrač 2 je pobedio rundu jer je uneo približniji broj ciljanom broju!'), nl;
            nl,write('Izjednačenje! Oba igrača su podjednako blizu.'), nl
        );
        Result1 =:= 0 ->
        nl,write('Igrač 1 je uneo nepravilni izraz, a igrač 2 je pobedio zbog pravilnog izraz.'), nl,
        add_score(2, 5);  % Igrač 2 dobija 5 poena
        Result2 =:= 0 ->
        nl,write('Igrač 2 je uneo nepravilni izraz, a igrač 1 pobedio zbog pravilnog izraz.'), nl,
        add_score(1, 5)  % Igrač 1 dobija 5 poena
    ).
```

Ovaj deo koda definiše logiku za proveru pobednika u igri "Moj Broj" na osnovu rezultata koje su igrači postigli.

1. **check_winner/3**:
    - Prihvaća tri argumenta: Result1 (rezultat igrača 1), Result2 (rezultat igrača 2) i Target (ciljni broj).
2. **Proverava različite uslove**:
    - **Oba igrača pogode tačan broj**: Ako su oba rezultata jednaka cilju, obojica dobijaju po 5 poena, a ispisuje se poruka o tome.
    - **Igrač 1 pogodi tačan broj**: Ako samo igrač 1 pogodi, dobija 10 poena i ispisuje se odgovarajuća poruka.
    - **Igrač 2 pogodi tačan broj**: Ako samo igrač 2 pogodi, dobija 10 poena i ispisuje se odgovarajuća poruka.
    - **Oba igrača unesu nevalidne izraze**: Ako su oba rezultata 0, ispisuje se poruka o nevalidnim izrazima.
    - **Oba igrača unesu validne izraze**: U ovom slučaju se izračunavaju razlike između rezultata i ciljnog broja (Diff1 i Diff2).
        - **Igrač 1 bliži cilju**: Ako je razlika igrača 1 manja, dobija 5 poena i ispisuje se odgovarajuća poruka.

        - **Igrač 2 bliži cilju**: Ako je razlika igrača 2 manja, dobija 5 poena i ispisuje se odgovarajuća poruka.
        - **Izjednačenje**: Ako su razlike jednake, ispisuje se poruka o izjednačenju.
    - **Igrač 1 unosi nevalidan izraz**: Ako je rezultat igrača 1 0, a igrač 2 unosi validan izraz, igrač 2 dobija 5 poena i ispisuje se odgovarajuća poruka.
    - **Igrač 2 unosi nevalidan izraz**: Slično kao prethodni slučaj, ako je rezultat igrača 2 0, igrač 1 dobija 5 poena.

Ova logika omogućava pravednu procenu rezultata i dodelu poena na osnovu uspešnosti igrača u rundi.

```
156  % Ažuriranje poena igrača
157  add_score(Player, Points) :-
158      retract(scores(Score1, Score2)),
159      (
160          Player == 1 -> NewScore1 is Score1 + Points, NewScore2 = Score2;
161          Player == 2 -> NewScore1 = Score1, NewScore2 is Score2 + Points
162      ),
163      assert(scores(NewScore1, NewScore2)).
164
165  % Ispis trenutnih poena nakon svake runde
166  print_scores :-
167      scores(Score1, Score2),
168      nl,
169      write('Trenutni poeni: '), nl,
170      write('Igrač 1: '), write(Score1), nl,
171      write('Igrač 2: '), write(Score2), nl.
```

Ovaj deo koda se bavi ažuriranjem i ispisom trenutnih poena igrača u igri "Moj Broj".

1. **add_score/2**:
    - Ova funkcija ažurira poene igrača.
    - **Argumenti**:
        - Player: Igrač čiji se poeni ažuriraju (1 ili 2).
        - Points: Broj poena koji se dodaju.
    - **Logika**:
        - Prvo se koristi retract/1 da bi se uklonila trenutna vrednost poena za oba igrača iz baze podataka (predikata scores/2).
        - Na osnovu toga koji igrač je prosledjen kao argument (Player), izračunava se novi rezultat za odgovarajućeg igrača:

        - Ako je Player 1, dodaju se poeni igraču 1, a rezultat igrača 2 ostaje nepromenjen.
        - Ako je Player 2, dodaju se poeni igraču 2, a rezultat igrača 1 ostaje nepromenjen.
    - Na kraju, novi rezultati se čuvaju pomoću assert/1 tako što se ponovno dodaje predikat scores/2 sa novim vrednostima.

2. **print_scores/0**:
    - Ova funkcija ispisuje trenutne poene oba igrača.
    - **Logika**:
        - Prvo se poziva predikat scores/2 kako bi se dobili trenutni poeni igrača 1 i igrača 2.
        - Zatim se ispisuju trenutni poeni na konzoli, uključujući poruke koje označavaju koji igrač ima koji broj poena.

Ove funkcije omogućavaju praćenje i prikaz trenutnog stanja poena tokom igre, čime se poboljšava korisničko iskustvo i omogućava igračima da prate svoj napredak.

```
172  % Pitanje da li igrači žele da nastave ili završe igru
173  continue_game :-
174      nl,write('Da li želite da završite igru? (da/ne): '), nl,nl,
175      read(Response),
176      (   Response == da ->
177          print_scores,  % Ispis finalnih poena
178          nl,
179          declare_winner,  % Provera ko je pobednik
180          nl,
181          write('Kraj igre!')
182      ;   Response == ne ->
183          play_game
184      ).
```

Ova funkcija se bavi pitanjem da li igrači žele da nastave sa igrom ili da je završe. Funkcija **continue_game/0** sadrži:

1. **Interakcija sa igračima**:
    - Prvo, ispisuje poruku koja pita igrače da li žele da završe igru.
    - Očekuje unos od strane igrača, koji može biti da ili ne.
2. **Logika na osnovu odgovora**:
    - **Ako je odgovor da**:
        - Poziva print_scores/0 da bi ispisao finalne poene oba igrača.

        - Poziva declare_winner/0 da proveri ko je pobednik igre na osnovu konačnih rezultata.
        - Zatvara igru sa porukom Kraj igre!.
    - **Ako je odgovor ne**:
        - Poziva play_game/0 da započne novu rundu igre.

Ova funkcija omogućava igračima da donesu odluku o tome da li žele da nastave ili završe igru, čime se pruža fleksibilnost u toku igre. Takođe, ona obezbeđuje da se konačni rezultati prikažu pre nego što igra završi, što je važno za zatvaranje igre na pozitivan način.

```
186  % Deklariše pobednika na osnovu rezultata
187  declare_winner :-
188      scores(Score1, Score2),
189      (   Score1 > Score2 ->
190          nl, write('Igrač 1 je pobednik sa osvojenih'), write(Score1), write(' poena!'), nl
191      ;   Score2 > Score1 ->
192          nl, write('Igrač 2 je pobednik sa osvojenih'), write(Score2), write(' poena!'), nl
193      ;   nl, write('Igra je završena nerešeno!'), nl
194      ).
```

Ova funkcija služi za deklasiranje pobednika igre na osnovu trenutnih rezultata igrača.

Prvo, koristi predikat scores/2 da dobije trenutne poene igrača 1 i igrača 2. Zatim, koristi uslovne izraze da uporedi rezultate:

- Ako su poeni igrača 1 veći od poena igrača 2, ispisuje poruku da je igrač 1 pobednik, zajedno sa njegovim ukupnim brojem poena.
- Ako su poeni igrača 2 veći od poena igrača 1, ispisuje poruku da je igrač 2 pobednik, uz njegov broj poena.
- Ako su poeni izjednačeni, ispisuje poruku da je igra završena nerešeno.

Ova funkcija pruža jasan pregled konačnog ishoda igre, obezbeđujući da se svi igrači obaveste o rezultatima na kraju takmičenja.

4. Primeri različitih slučajeva toka igre

```
?- [mojBroj].
true.

?- play_game.
Dobrodošli u igru Moj Broj!

Ciljni broj: 487
Dostupni brojevi: [2,8,6,3,14,78]

Igrač 1 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: 78*6+3.

Igrač 1 je dobio broj: 471

Igrač 2 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: (78*6)+14.

Igrač 2 je dobio broj: 482


Igrač 2 je pobedio rundu jer je uneo približniji broj ciljanom broju!

Trenutni poeni:
Igrač 1: 0
Igrač 2: 5

Da li želite da završite igru? (da/ne):

|: ne.
```

```
Ciljni broj: 570
Dostupni brojevi: [9,6,9,3,13,94]

Igrač 1 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: 9*6*6.

Uneseni izraz igrača 1 nije validan ili su brojevi korišćeni više puta nego što je dostupno.

Igrač 2 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: 9*6/13.

Uneseni izraz igrača 2 nije validan ili su brojevi korišćeni više puta nego što je dostupno.


Oba igrača su uneli nepravilne izraze.
```

```
Trenutni poeni:
Igrač 1: 5
Igrač 2: 5

Da li želite da završite igru? (da/ne):

|: ne.
Dobrodošli u igru Moj Broj!

Ciljni broj: 729
Dostupni brojevi: [4,1,2,3,37,73]

Igrač 1 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: 73*(4+3+2+1)-1.

Uneseni izraz igrača 1 nije validan ili su brojevi korišćeni više puta nego što je dostupno.

Igrač 2 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: (4+3+2+1)*73.

Igrač 2 je dobio broj: 730


Igrač 1 je uneo nepravilni izraz, a igrač 2 je pobedio zbog pravilnog izraz.

Trenutni poeni:
Igrač 1: 5
Igrač 2: 10
```

```
Ciljni broj: 246
Dostupni brojevi: [7,5,4,4,39,61]

Igrač 1 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: 61*4+7-5.

Igrač 1 je dobio broj: 246

Igrač 2 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: (7-5)+(4*61).

Igrač 2 je dobio broj: 246


Oba igrača su pogodila tačan broj!

Trenutni poeni:
Igrač 1: 5
Igrač 2: 5
```

```
?- play_game.
Dobrodošli u igru Moj Broj!

Ciljni broj: 501
Dostupni brojevi: [7,4,3,5,47,51]

Igrač 1 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: 51*(7+3)-4-5.

Igrač 1 je dobio broj: 501

Igrač 2 je na potezu! Unesite izraz koristeći dostupne brojeve i osnovne matematičke operacije(Celi brojevi su samo validni!):
|: 51*7.

Igrač 2 je dobio broj: 357


Igrač 1 je pogodio tačan broj!

Trenutni poeni:
Igrač 1: 10
Igrač 2: 0

Da li želite da završite igru? (da/ne):

|: da.

Trenutni poeni:
Igrač 1: 10
Igrač 2: 0


Igrač 1 je pobednik sa osvojenih10 poena!

Kraj igre!
true.
```