Program 5.3: Translator języka

```
program analizatorogólny;
label 99;
const pusty = '*';
type wskaźnik = ↑węzeł;
    poczwsk = ↑początek;
    węzeł = record nast, człalt : wskaźnik;
                case końcowy : boolean of
                    true: (ksym: char);
                    false: (psym: poczwsk)
                end;
    początek = record sym: char;
                    wejście: wskaźnik;
                    nast : poczwsk
                end;
var lista, wartownik, h: poczwsk;
    p: wskaźnik;
    sym: char;
    ok: boolean;

procedure pobsym;
begin
    repeat read(sym); write(sym) until sym ≠ ' '
end {pobsym};

procedure znajdź(s: char; var h: poczwsk);
{zlokalizuj na liście symbol pomocniczy s; jeśli jest nieobecny, to go dodaj}
    var h1 : poczwsk;
begin h1 := lista; wartownik↑.sym := s;
    while h1↑.sym ≠ s do h1 := h1↑.nast;
    if h1 = wartownik then
        begin {wstaw} new(wartownik);
            h1↑.nast := wartownik; h1↑.wejście := nil
        end;
    h := h1
end {znajdź};

procedure błąd;
begin writeln:
    writeln('NIEPOPRAWNA SKŁADANIA'); goto 99
end {błąd};

procedure składnik(var p, q, r: wskaźnik);
    var a, b, c : wskaźnik;
    procedure czynnik(var p, q: wskaźnik);
        var a, b: wskaźnik; h: poczwsk;
    begin if sym in ['A'..'Z', pusty] then
        begin {symbol} new(a);
            if sym in ['A'. .'H'] then
            begin {pomocniczy} znajdź(sym, h);
                a↑.końcowy := false; a↑.psym := h
        end else
        begin {końcowy}
            a↑.końcowy := true; a↑.ksym := sym
        end;
        p := a; q := a; pobsym
        end else
```

```
        if sym = '[' then
        begin pobsym; skladnik(p, a, b); b↑.nast := p;
            new(b); b↑.końcowy := true; b↑.ksym := pusty;
            a↑.człalt := b; q := b;
            if sym = ' ]' then pobsym else błąd
        end else błąd
    end {czynnik};
begin czynnik(p, a); q := a;
    while sym in ['A'..'Z',  '[', pusty] do
    begin czynnik(a↑.nast, b); b↑.człalt := nil;  a := b
    end;
    r := a
end {składnik};

procedure wyrażenie(var p, q: wskaźnik);
    var a, b, c : wskaźnik;
begin składnik(p, a, c); c↑.nast := nil;
    while sym = ', ' do
    begin pobsym;
        składnik(a.↑człalt, b, c); c↑.nast := nil; a := b
    end;
    q:=a
end {wyrażenie};

procedure analizuj(cel: poczwsk; var jest: boolean);
    var s: wskaźnik;
begin s := cel↑.wejście;
    repeat
        if s↑.końcowy then
        begin if s↑.ksym := sym then
                    begin jest := true; pobsym
                    end

                else jest := (s↑.ksym=pusty)
        end
        else analizuj(s↑.psym, jest);
        if jest then s := s↑.nast else s := s↑.czlalt
    until s = nil
end {analizuj }

begin {produkcje} {program główny}
    pobsym; new(wartownik); lista := wartownik;
    while sym  ≠ '$' do
    begin znajdź(sym, h);
        pobsym; if sym= '=' then pobsym else błąd;
        wyrażenie(h↑.wejście, p);  p↑.człalt := nil;
        if sym ≠ '.' then błąd:
        writeln; readln; pobsym
    end;
    {sprawdź czy wszystkie symbole są zdefiniowane}
    h := lista; ok := true;
    while h ≠ wartownik do
    begin if h↑.wejście = nil then
            begin  writeln('SYMBOL NIEZDEFINIOWANY', h↑. sym);
                ok := false
            end;
        h := h↑.nast.
    end;
```

```
    if  not ok then goto 99;
    {symbol początkowy gramatyki}
    pobsym; znajdź(sym, h); readln; writeln;
    {zdania}
    while not  eof(input) do
        begin write('  '); pobsym; analizuj (h, ok);
            if ok and (sym = '.')   then writeln('POPRAWNE')
                                    else writeln('NIEPOPRAWNE');
            readln
        end;
99: end
```