Program 5.6. Kompilator języka PL/0

```
program PL/0 (input, output ) ;
{kompilator PL/0 łącznie z generowaniem kodu wynikowego}
label 99 ;
const lsz = 11;     {liczba słów zastrzeżonych}
    tmax = 100;   {długość tablicy identyfikatorów}
    cmax = 14;    {maksymalna liczba cyfr w liczbach}
    a l= 10;            {długość identyfikatorów}
    amax = 2047; {maksymalny adres}
    pozmax = 3 ;  {maksymalna głębokość zagnieżdżenia bloków}
    rmax=200;     {długość tablicy z rozkazami kodu wynikowego}
type symbol =
    (żaden, idem, liczba, plus, minus, razy, dziel, oddsym, równe, różne, mniejsze, mnrówne, większe, wrówne,
    lnaw, pnaw, przec, średnik, kropka, przypis, beginsym, endsym, ifsym, thensym, whilesym, dosym, callsym,
    constsym, varsym, procsym);
    alfa=packed array [1..al] of char;
    obiekt = (stała, zmienna, procedura);
    zbiórsym = set of symbol;
    czynności = (sta, opr, ład, pam, pro, prz, skb, skw);
    rozkaz = packed record
                cz : czynności;   {kod czynności}
                p  : 0. .pozmax;  {poziom}
                a  : 0. .amax;          {przesunięcie}
             end;
      {STA  0,  a : ładuj stałą a
       OPR   0,  a : wykonaj operację a
       ŁAD   p,  a : ładuj zmienną p, a
       PAM   p,  a : pamiętaj zmienną p, a
       PRO   p,  a : wywołaj procedurę a na poziomie p
       PRZ   0,  a : zwiększ t-rejestr o a
       SKB   0,  a : skocz bezwarunkowo do a
       SKW   0,  a : skocz warunkowo do a}
var  ch: char;          {ostatni wczytany znak}
    sym: symbol;        {ostatni wczytany symbol }
    id: alfa;           {ostatni wczytany identyfikator}
    num: integer;           {ostatnia wczytana liczba}
    lz: integer;        {licznik znaków}
    dl: integer;        {długość linii}
    kk: integer;
    lr: integer;        {licznik rozkazów}
    linia: array. . [1','81] of char;
    a: alfa;
    kod: array [0. .rmax] of rozkaz;
    słowo: array [l. .lsz] of alfa;
    zsym: array [1. .lsz] of symbol;
    ssyrn: array [char] of symbol;
    mnem: array [czynności] of
        packed array [1. .5] of char;
    poczdekl, poczinstr, poczczyn: zbiórsym;
    tablica:  array [0. .tmax] of
              record nazwa: alfa;
                case rodzaj: obiekt of
                stała: (wartość: integer);
                zmienna, procedura: (poziom, adr: integer)
              end;
    błędy: boolean;
```

```
procedure błąd(n: integer);
begin writeln(' **** ', '  ': lz–1, '↑', n: 2); błędy :=true
end {błąd};

procedure pobsym;
    var i, j, k: integer;

    procedure pobznak;
    begin  if lz =dl then
        begin if eof(input) then
                begin write(' PROGRAM NIEDOKOŃCZONY'); goto 99
                end;
            dl := 0; lz := 0; write(lr: 5, ' ');
            while ¬ eoln(input) do
                begin dl := dl + l; read(ch); write(ch); linia[dl] := ch
                end;
            writeln; dl := dl + 1; read(linia[dl])
        end;
        lz := lz + 1; ch := linia[lz]
    end {pobznak};

begin {pobsym}
    while ch = '  ' do pobznak;
    if ch in ['A'. .'Z'] then
    begin {identyfikator lub słowo zastrzeżone} k := 0;
        repeat if k < al then
            begin k := k+1; a[k] := ch
            end;
            pobznak
        until ¬ (ch in ['A'. . 'Z', '0'. . '0'];
        if k ≥ kk then kk := k else
            repeat a[kk] := '  '; kk := kk – 1
            until kk = k;
        id := a; i := 1; j := lsz;
        repeat: k := (i + j) div 2;
            if id ≤ słowo[k] then j := k – 1;
            if id ≥ słowo[k] then i := k + 1
        until i > j;
        if i – 1 > j then sym := zsym[k] else sym := ident
    end else
    if ch in ['0'. . '9'] then
    begin {liczba} k := 0; num := 0; sym := liczba;
        repeat num := 10*num+(ord(ch)–ord('0'));
            k := k+1; pobznak
        until ¬ (ch in ['0'. . '9']);
        if k > cmax then błąd(30)
    end else
    if ch = ': ' then
    begin pobznak;
        if ch='=' then
        begin sym „= przypis; pobznak
        end else := żaden;
    end else
    begin sym := ssym[ch]; pobznak
    end
end {pobsym};
```

```
procedure gen(x: czynności; y, z: integer);
begin if lr > rmax then
        begin write(' ZA DŁUGI PROGRAM'); goto 99
        end;
    with kod[lr] do
        begin cz := x; p := y; a := z
        end;
    lr := lr+1
end {gen};
procedure test(s), s2: zbiórsym ; n: integer);
begin if ¬ (sym in s1) then
        begin błąd(n); s1:= s1+s2;
            while ¬ (sym in s1) do pobsym
        end
end {test};

procedure blok(poz, tx: integer; fpocz: zbiórsym);
    var     dx: integer;        {indeks segmentu danych}
        tx0: integer;       {początkowa wartość indeksu tx}
        lr0: integer;       {początkowa wartość licznika lr}
    procedure wprowadź(k: obiekt);
    begin {wprowadź obiekt do tablicy}
        tx := tx+1;
        with tablica[tx] do
        begin nazwa := id; rodzaj := k;
            case k of
                stała: begin if num > amax then
                                begin błąd(31); num := 0 end;
                                    wartość := num
                                end;
                zmienna: begin poziom := poz; adr :=dx; dx := dx+1;
                            end;
                procedura: poziom := poz
                end
        end
    end {wprowadź};
    function pozycja(id: alfa): integer;
        var i: integer;
    begin {znajdź identyfikator id w tablicy}
        tablica[0].nazwa := id; i := tx;
        while tablica[i].nazwa ≠ id do i := i –1;
        pozycja := i
    end {pozycja};

    procedure deklstałej;
    begin if sym = ident then
        begin pobsym;
            if sym in [równe, przypis) then
            begin if sym = przypis then błąd(1);
                pobsym;
                if sym = liczba then
                    begin wprowadź(stała); pobsym
                    end
                else błąd(2)
            end else błąd(3)
        end else błąd(4)
    end {deklstałej};
```

```
procedure deklzmiennej;
begin if sym = ident then
        begin wprowadź(zmienna); pobsym
        end else błąd(4)
end {deklzmiennej};

procedure drukujkod;
    var i: integer;
begin {drukuj program wyprodukowany dla aktualnego bloku}
    for i := lr0 to lr – 1 do
        with kod[i] do
            writeln(i, mnem[cz]: 5, p: 3, a: 5)
end {drukujkod};

procedure instrukcja(fpocz: zbiórsym);
    var i, lr1, lr2: integer;
    procedure wyrażenie(fpocz: zbiórsym);
        var opaddyt : symbol;
        procedure składnik(fpocz: zbiórsym);
            var opmult: symbol;
            procedure czynnik(fpocz: zbiórsym);
                var i: integer;
            begin test(poczczyn, fpocz, 24);
                while sym in poczczyn do
                begin
                    if sym = ident then
                    begin i := pozycja(id);
                        if i = 0 then błąd(11) else
                        with tablica[i] do
                        case rodzaj of
                            stała: gen(sta, 0, wartość);
                            zmienna: gen(ład, poz–poziom, adr);
                            procedura: błąd(21)
                        end;
                        pobsym
                        end else
                        if sym = liczba then
                        begin if num > amax then
                                begin błąd(31); num := 0
                                end;
                            gen(sta, 0, num); pobsym
                        end else
                        if sym = lnaw then
                        begin pobsym; wyrażenie([pnaw] + fpocz);
                            if sym = pnaw then pobsym else błąd(22)
                        end;
                        test(fpocz, [lnaw], 23)
                    end
                end {czynnik};
            begin {składnik} czynnik(fpocz + [razy, dziel]);
                while sym in [razy, dziel] do
                    begin opmult := sym; pobsym;
                        czynnik(fpocz + [razy, dziel]);
                        if opmult = razy then gen(opr, 0, 4)
                        else gen(opr, 0, 5)
                    end
```

```
        end {składnik};
        begin {wyrażenie}
        if sym in [plus, minus] then
            begin opaddyt := sym; pobsym;
                składnik(fpocz + [plus, minus]);
                if opaddyt = minus then gen(i, 0, 1)
            end else składnik(fpocz + [plus, minus]);
        while sym in [plus, minus] do
            begin opaddyt := sym; pobsym;
                składnik(fpocz + [plus, minus]);
                if opaddyt = plus then gen(opr, 0, 2) else gen(opr, 0, 3)
            end
    end {wyrażenie};

    procedure warunek(fpocz: zbiórsym);
        var oprel: symbol;
    begin
        if sym = oddsym then
        begin pobsym; wyrażenie(fpocz); gen(opr, 0, .6)
        end else
        begin wyrażenie([równe, różne, mniejsze, większe, mnrówne, wrówne] + fpocz);
            if ¬ (sym in [równe, różne, mniejsze, większe, mnrówne, wrówne]) then błąd(20) else
            begin oprel := sym; pobsym; wyrażenie(fpocz);
                case oprel of
                    równe:    gen(opr, 0, 8);
                    różne:    gen(opr, 0, 9);
                    mniejsze: gen(opr, 0, 10);
                    wrówne:   gen(opr, 0, 11);
                    większe:  gen(opr, 0, 12);
                    mnrówne: gen(opr, 0, 13);
                end
            end
        end
    end {warunek};
begin {instrukcja}
    if sym = ident  then
    begin i := pozycja(id);
        if i = 0 then błąd(11) else
        if tablica[i].rodzaj ≠ zmienna then
            begin {przypisanie wartości nie zmiennej} błąd(12); i := 0
            end;
        pobsym; if sym = przypis then pobsym else błąd(13);
        wyrażenie(fpocz) ;
        if i ≠ 0 then
            with tablica[i] do gen(pam, poz-poziom, adr)
    end else
    if sym = callsym then
    begin pobsym;
        if sym ≠ ident then błąd(14) else
            begin i := pozycja(id);
                if i= 0 then błąd(11) else
                with tablica[i] do
                    if rodzaj = procedura then gen(pro, poz-poziom, adr)
                    else błąd(15);
                pobsym
            end
    end else
    if sym = ifsym then
```

```
        begin pobsym; warunek([thensym, dosym] +fpocz);
            if sym = thensym then pobsym else błąd(16);
            lr 1 := lr; gen(skw, 0, 0);
            instrukcja(fpocz); kod[lr1].a := lr
        end else
        if sym = beginsym then
        begin pobsym; instrukcja([średnik, endsym]+fpocz);
            while sym in [średnik] + poczinstr do
            begin
                if sym = średnik then pobsym else błąd(10);
                instrukcja([średnik, endsym] + fpocz)
            end;
            if sym = endsym then pobsym else błąd(17)
        end else
        if sym = whilesym then
        begin lr1 := lr; pobsym; warunek([dosym]+fpocz);
            lr2 := lr; gen(skw, 0, 0);
            if sym=dosym then pobsym else błąd(18);
            instrukcja(fpocz); gen(skb, 0,lr1); kod[lr2].a := lr
        end;
        test(fpocz, [ ], 19)
    end [instrukcja};

begin [blok] dx := 3 ; tx0 == tx;
    tablica[tx].adr := lr; gen(skb, 0, 0);
    if poz > pozmax then błąd(32);
    repeat
        if sym = constsym then
        begin pobsym;
            repeat deklstałej;
                while sym = przec do
                    begin pobsym ; deklstałej
                    end;
                if sym = średnik then pobsym else błąd(5)
            until sym ≠ ident
        end;
        if sym = varsym then
        begin pobsym;
            repeat deklzmiennej;
                while sym = przec do
                    begin pobsym; deklzmiennej
                    end;
                if sym = średnik then pobsym else błąd(5)
            until sym ≠ ident;
        end;
        while sym = procsym do
        begin pobsym;
            if sym = ident then
                begin wprowadź(procedura); pobsym
                end
            else błąd(4);
            if sym = średnik then pobsym else błąd(5);
            blok(poz + 1, tx, [średnik] + fpocz);
            if sym = średnik then
                begin pobsym; test(poczinstr + [ident, procsym], fpocz, 6)
                end
            else błąd(5)
        end;
```

```
        test(poczinstr + [ident], poczdekl, 7)
    until ¬ (sym in poczdekl);
    kod [tablica[tx0].adr].a := lr;
    with tablica[tx0] do
        begin adr := lr; {adres początkowy kodu}
        end;
    lr0 := lr; gen(alo, 0, dx);
    instrukcja([średnik, endsym] + fpocz);
    gen(opr, 0, 0); {powrót}
    test(fpocz, [ ], 8);
    drukujkod;
end {blok};

procedure interpretuj;
    const długstosu = 500
    var p, b, t: integer; {rejestry programowy, bazowy, stosowy}
        i: rozkaz; {rejestr rozkazu}
        s: array[l..długstosu] of integer; {stos — pamięć z danymi}
    function baza(p: integer): integer;
        var b1: integer;
    begin b1 := b; {znajdź bazę segmentu leżącego p poziomów niżej}
        while p > 0 do
            begin b1 := s[b1]; p := p–1
            end;
        baza := b1
    end {baza};

begin writeln('POCZĄTEK INTERPETAC]I');
    t := 0; b := 1; p := 0;
    s[1] := 0; s[2] := 0; s[3] := 0;
    repeat i := kod[p]; p := p+1;
        case i.cz of
      sta:  begin t := t:1; s[t] := i.a
            end;
      opr:  case i.a of {operator}
            0:  begin {powrót}
                    t := b – 1; p := s[t+3]; b := s[t+2];
                end;
            l:  s[t] := –s[t];
            2:  begin t := t – 1; s[t] := s[t]+s[t+1]
                end;
            3:  begin t := t – l; s[t] := s[t] – s[t+1]
                end;
            4:  begin t := t – l; s[t] := s[t]*s[t+1]
                end;
            5:  begin t := t – 1; s[t] := s[t] div s[t+1]
                end;
            6:  s[t] := ord(odd(s[t]));
            8:  begin t := t – 1; s[t] := ord(s[t]=s[t+1])
                end;
            9:  begin t := t – 1; s[t] := ord(s[t]≠s(t+1])
                end;
           10:  begin t := t – 1; s[t] := ord(s[t] < s[t+1])
                end;
           11:  begin t := t – 1; s[t] := ord(s[t] ≥ s[t])
                end;
           12:  begin t := t – l; s[t] := ord(s[t] > s[t+1])
                end;
```

```
            13:  begin t := t – 1; s[t] := ord(s[t] ≤ s[t+1])
                 end;
          end;
        ład:     begin t := t+l; s[t] := s[baza(i.p)+i.a]
                 end;
        pam:     begin s[baza(i.p)+i.a] := s[t] : writeln(s [t]) ; t := t – 1
                 end;
        pro:     begin {nowy blok}
                   s[t+1] := baza(i.p); s[t+2] := b; s[t+3] := p;
                   b := t+1 ; p := i.a
                 end;
        prz:     t := t + i.a;
        skb:     p := i.a;
        skw:     begin if s[t] = 0 tben p := i.a; t := t – 1
                 end
      end {case}
  until p = 0;
  write('KONIEC INTERPRETACJI');
end {interpretuj};

begin {program główny}
  for ch := 'A' to ';' do  ssym[ch] := żaden;
  słowo[1] := 'BEGIN     ';   słowo[2] := 'CALL          ';
  słowo[3j := 'CONST     ';   słowo[4] := 'DO            ';
  słowo[5] := 'END     ';   słowo[6) := 'IF          ';
  słowo[7] := 'ODD       ';   słowo[8] := 'PROCEDURE     ';
  słowo[9] := 'THEN      ';   słowo[10] := 'VAR          ';
  słowo[11] := 'WHILE    ';
  zsym[1] := beginsym;      zsym[2] := callsym;
  zsym[3] := constsym;      zsym[4] := dosym;
  zsym[5] := endsym;        zsym[6] := ifsym;
  zsym[7] := oddsym;        zsym[8] := procsym;
  zsym[9] := thensym;       zsym[10] := varsym;
  zsym[11] := whilesym;
  ssym['+'] := plus;        ssym['–') := minus;
  ssym['*'] := razy;        ssym['/') := dziel;
  ssym['('] := lnaw;        ssym[")] := pnaw;
  ssym['='] := równe;       ssym[','] := przec;
  ssym['.'] := kropka;      ssym[' ~']  := różne;
  ssym['<'] := mniejsze;    ssym['>'] := większe;
  ssym['≤'] := mnrówne;     ssym['≥'] := wrówne;
  ssym[';'] := średnik;
  mnem[sta]' := STA';       mnem[opr] := 'OPR';
  mnem(ład] := 'Ł.AD';      mnem[pam] := 'PAM';
  mnem[pro] := 'PRO';       mnem[alo] := 'PRZ';
  mnem[skb] := 'SKB';       mnem[skw] := 'SKW';
  poczdekl := (constsym, varsym, procsyrn];
  poczinstr := [beginsym, callsym, ifsym, whilesym];
  poczczyn := [idem, liczba, lnaw];
  page(output); błędy := false;
  lz := 0; lr := 0; dl := 0; ch := ' ';  kk := al; pobsym;
  blok(0, 0, [kropka] + poczdekl + poczinstr);
  if sym ≠ kropka then błąd(9);
  if błędy then write('BŁĘDY W PROGRAMIE PL/0') else interpretuj;
99: writeln;
end.
```