ЛАБАРАТОРНАЯ РАБОТА № 3

Цыклы

Мэта: азнаёміць з асноўнымі канструкцыямі, з дапамогай якіх рэалізоўваюцца цыклічныя алгарытмы на мове Паскаль.

Алгарытм называецца цыклічным, калі ён змяшчае шматразовае выкананне адной і той жа галіны пры розных значэннях прамежкавых даных. Шматразова выконваемы ўчастак вылічэнняў называюць целам цыкла, а пераменную, якая змяняецца ў цыкле і вызначае колькасць паўтораў, – параметрам цыкла. Алгарытм цыклічнай структуры ў агульным выглядзе павінен змяшчаць:

– падрыхтоўку цыкла – заданне пачатковых значэнняў параметра цыкла перад яго выкананнем;

– цела цыкла – дзеянні, якія паўтараюцца ў цыкле для розных значэнняў параметра цыкла;

– мадыфікацыю (змяненне) значэнняў параметра цыкла перад кожным новым яго паўтарэннем;

– кіраванне цыклам – праверку ўмовы працягу (ці заканчэння) цыкла і пераход на пачатак цела цыкла, калі выконваецца ўмова працягу цыкла (ці выхад з цыкла, калі выконваецца ўмова заканчэння цыкла).

Колькасць паўтораў цыкла можа быць зададзена ў яўнай ці няяўнай форме. Для праграмаваня цыклічных алгарытмаў у Паскалі існуюць тры віды аператараў арганізацыі цыклаў:

1) *while* – аператар цыкла з прадумовай;

2) *repeat* – аператар цыкла з постумовай;

3) *for* – аператар цыкла з кіруючым параметрам.

Аператар цыкла з параметрам *for* мае два віды:

for i:=N1 to N2 do аператар;

for i:=N1 downto N2 do аператар;

i – пераменная цыкла цэлага, сімвальнага, адрэзачнага тыпаў ці пералічваемага тыпу, $N1$, $N2$ – пачатковае і канечнае значэнні пераменнай i.

Першая форма выкарыстоўваецца для запісу цыкла па ўзрастаючым значэнням параметра, другая – па ўбываючым.

З а ў в а г і .

1. Унутры цыкла нельга змяняць значэнні $N1$, $N2$ і пераменную цыкла i.

2. У аператары *for* не дапускаецца змяненне параметра на велічыню, якая не роўная 1.

Адвольны крок змянення параметра можна арганізаваць з дапамогай аператараў *repeat* ці *while*.

Прыклад 1. Вызначым найбольшы з адмоўных элементаў паслядоўнасці $a_n = \cos(i^2)+1$:

```
program ssss;
    var k,i,n:integer;
        a,m:real;
BEGIN
    write('увядзіце размернасць n? '); readln(n);
    k:=0; m:=0;
    for i:=1 to n do
          begin
                a:=cos(i*i)+1;
                if a<0 then k:=k+1;
                if k=1 then begin
                                  m:=a;
                                  k:=3
                            end;
                if (a<0) and (a>m) then m:=a;
          end;
    if k<>0 then writeln ( 'адказ ',m)
             else writeln ('адмоўных няма')
END.
```

Прыклад 2. Вызначым колькасць элементаў паслядоўнасці цэлых лікаў, якія меншыя сярэдняга арыфметычнага элементаў папярэдняга і наступнага дадзенаму элементу паслядоўнасці.

```
program main;
    var n, i, R: Integer;
        a,x,y: Integer;
BEGIN
    R:=0;
    Write('увядзіце колькасць элементаў паслядоўнасці');
    ReadLn (n);
    WriteLn ('увядзіце 1 элемент'); readln(x);
    writeln('увядзіце 2 элемент'); readln(a);
    for i:=3 to n do
```

```
        begin
            writeln('увядзіце ‘,i,’ элемент');  ReadLn (y);
            if a < (x+y)/2 then R:= R+1;
            x:=a; a:=y;
        end;
    writeLn (R);
END.
```

Аператар цыкла з прадумовай:

while умова U do аператар;

Калі вынік праверкі ўмовы *U* з’яўляецца (*True*) праўдзівым, то выконваецца аператар цыкла. Калі ж лагічны выраз *U* ніколі не прыме значэнне *False* (ложна), то цыкл будзе выконвацца бясконцую колькасць разоў. Такую сітуацыю звычайна называюць "зацыкленнем".

Прыклад 3. Вызначым колькасць лічбаў дадзенага натуральнага ліку і яго першую лічбу.

```
program cifr;
    var c,k:integer;n:longint;
BEGIN
    write(‘увядзіце натуральны лік’);readln(n);
    K:=0;
    while n<>0 do
        begin
            c:=n mod 10;
            n:=n div 10;
            k:=k+1;
        end;
    writeln(‘k=’,k,’ c=’,c);
END.
```

Пры такім рашэнii задачы мы абмежаваныя тым, што максімальна дапушчальны для ўводу лік 2147483647. Разгледзім другі варыянт рашэння гэтай задачы:

```
program cifr;
    var n,c:real;k:integer;
BEGIN
    write('увядзіце натуральны лік');readln(n);
    k:=0;
    While n<>0 do
        begin
            c:=int(frac(n/10)*10);
            n:=int(n/10);
```

```
            k:=k+1;
        end;
    writeln('k=',k,' c=',c:5:0);
END.
```

Колькасць лічбаў у запісе натуральнага ліку можна было адразу падлічыць па формуле $k = \lfloor \lg n \rfloor + 1$.

Аператар цыкла з постумовай:

```
repeat
    аператары
until U;
```

Выкананне цыкла ў гэтым выглядзе перарываецца ў выпадку праўдзівасці ўмовы U . Калі лагічны выраз U ніколі не прымае значэнне *True* (праўдзіва), то цыкл будзе выконвацца бясконца.

Ніжняя мяжа аператараў цела цыкла ясна абазначана словам *until*, таму няма неабходнасці заключаць гэтыя аператары ў аператарныя дужкі *begin-end*. У той жа час дадатковы запіс аператарных дужак не з’яўляецца памылкай.

Прыклад 4. Сума лічбаў двухзначнага ліку роўна 6. Калі да гэтага ліку дабавіць 18, то атрымаецца лік, запісаны тымі ж лічбамі, але ў адваротным парадку. Знойдзем гэты двузначны лік:

```
program sls;
    var a,b,n:integer;
BEGIN
    writeln('варыянт 1');
    for a:=1 to 6 do
        for b:=0 to 5 do
            if (a+b=6) and (a*10+b+18=b*10+a) then n:=a*10+b;
    writeln(n);
    writeln('варыянт 2');
    a:=1;
    while a<=6 do
        begin
            b:=0;
            while b<=5 do begin
                if (a+b=6) and (a*10+b+18=b*10+a) then n:=a*10+b;
                b:=b+1
                        end;
            a:=a+1
        end;
    writeln(n);
    writeln('варыянт 3');
```

```
a:=1;
repeat
    b:=0;
    repeat
        if (a+b=6) and (a*10+b+18=b*10+a) then n:=a*10+b;
        b:=b+1
    until b>6;
    a:=a+1
until a>5;
writeln(n)
END.
```

Прыклад 5. Дадзены сапраўдны лік x, падлічым прыблізнае значэнне бясконцай сумы:

$$\sum_{k=1}^{\infty} \frac{(-1)^{k+1} x^k}{k} = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots \qquad (0 < x \le 1).$$

Трэба ўлічваць, што патрэбны вынік будзе дасягнуты, калі модуль n-га элемента сумы будзе меншы зададзенай хібнасці ε:

```
program summa;
    var x,e,a,s,x:real; c,k:integer;
BEGIN
    write('увядзіце значэнне E '); readln(e);
    repeat
        write(увядзіце лік x, 0< x ≤ 1 ');
        readln(x);
    until abs(x)<=1;
    s:=0; k:=1; a:=x; t:=1; c:=1;
    while abs(a/k)>=e do
                    begin
                        t:=t*x; a:=c*t/k; s:=s+a; k:=k+1; c:=-c;
                    end;
    writeln;
    writeln(' s=',s:7:3);
END.
```

Прыклад 6. Выведзем на экран усе поўныя лікі ў дыяпазоне ад 2 да n (натуральны лік называецца поўным, калі ён роўны суме ўсіх сваіх дзельнікаў, акрамя самога сябе: 6=1+2+3). Трэба ўлічваць, што дзельнік не можа быць большым за палову самога ліку:

```
program sower;
```

```
    var n,i,j,m:integer;
BEGIN
    write('n=? '); readln(n);
    for i:=2 to n do
        begin
            m:=0;
            for j:=1 to i div 2 do
                if i mod j=0 then m:=m+j;
            if m=i then write(i:7);
        end; writeln;
END.
```

Прыклад 7. Складзём праграму праверкі ведаў табліцы множання:

```
program tablica;
    const k=10;
    var a,b,c,d,i:integer;
BEGIN
    randomize; clrscr;
    for i:=1 to k do
        begin
            a:=random(8)+2; b:=random(8)+2; c:=a*b;
            writeln('Чаму роўна',a,'*',b,' ?');
            write('Увядзіце адказ'); readln(d);
            if c=d then writeln ('Правільны адказ')
                else writeln('Памылка');
            writeln;
        end;
END.
```