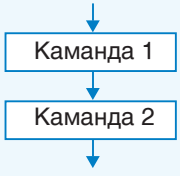
Глава 2
АЛГАРЫТМЫ АПРАЦОЎКІ РАДКОВЫХ ВЕЛІЧЫНЬ

§ 6. Асноўныя алгарытмічныя канструкцыі і тыпы даных

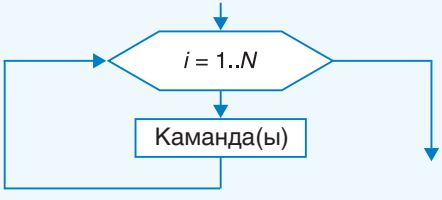
Прыклад 6.1. Блок-схемы алгарытмічных канструкцый.

1. Паслядоўнасць

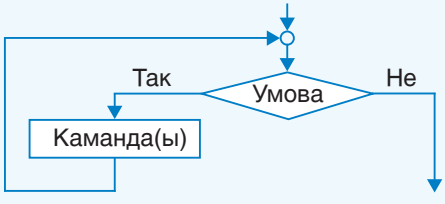


2. Цыкл

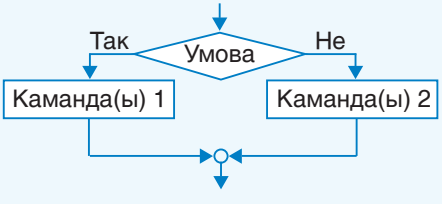
1) цыкл з параметрам



2) цыкл з перадумовай



3. Галінаванне



6.1. Асноўныя алгарытмічныя канструкцыі

Нагадаем некаторыя азначэнні, вядомыя вам з курсаў 7-га і 8-га класаў.

Алгарытм — канечная паслядоўнасць каманд, фармальнае выкананне якіх дазваляе атрымаць рашэнне задачы для любога дапушчальнага набору зыходных даных. Усе каманды падзяляюць на групы:

1. Каманды, якія непасрэдна выконваюцца ў праграме.

2. Каманды, якія змяняюць парадак выканання іншых каманд.

Любы алгарытм можа быць запісаны з выкарыстаннем базавых алгарытмічных канструкцый, а менавіта: **паслядоўнасць**, **паўтарэнне** і **галінаванне** (прыклад 6.1).

Праграма ўяўляе сабой запіс на некаторай фармальнай мове — мове праграмiравання. Камандамі ў мове праграмiравання лічаць:

- аператары (аператар прысвойвання, аператар галінавання, аператар цыкла і інш.);
- выклікі дапаможных алгарытмаў (убудаваных у бібліятэкі і створаных карыстальнікам).

Каманды цыкла і галінавання кіруюць парадкам выканання іншых каманд у праграме і належаць да **каманд**

кіравання (кіруючых канструкцый) (прыклад 6.2).

Аператар галінавання — каманда, якая рэалізуе алгарытмічную канструкцыю *галінаванне* на мове праграміравання.

Аператар цыкла — каманда, якая рэалізуе алгарытмічную канструкцыю *паўтарэнне* на мове праграміравання.

У Pascal існуюць розныя магчымасці кіраваць тым, колькі разоў будзе паўтарацца цела цыкла. Можа быць зададзена ўмова працягвання або заканчэння работы цыкла, а таксама колькасць паўтарэнняў цела цыкла.

Цыкл з перадумовай выкарыстоўваецца ў тым выпадку, калі вядома ўмова працягвання работы.

Цыкл з параметрам выкарыстоўваецца тады, калі вядома колькасць паўтарэнняў.

6.2. Дапаможныя алгарытмы

Дапаможны алгарытм — алгарытм, які можна выкарыстоўваць у іншых алгарытмах, запісаўшы яго імя і, калі неабходна, значэнні параметраў.

У мове Pascal выкарыстоўваюцца дапаможныя алгарытмы двух відаў: працэдуры і функцыі. Яны могуць быць з параметрамі або без параметраў (прыклад 6.3).

Апісанне працэдур і функцый паўтарае структуру праграмы на мове Pascal. Яно можа змяшчаць раздзел **var** для апісання пераменных, якія

Прыклад 6.2. Запіс аператараў галінавання і цыкла на мове Pascal.

Галінаванне ў поўнай форме

```
if <умова> then
begin
  каманды 1;
end
else
begin
  каманды 2;
end;
```

Галінаванне ў скарочанай форме

```
if <умова> then
begin
  каманды;
end;
```

Цыкл з перадумовай

```
while <умова> do
begin
  цела цыкла;
end;
```

Цыкл з параметрам

Параметр павялічваецца:

```
for var i := N1 to N2 do
begin
  цела цыкла;
end;
```

Параметр памяншаецца:

```
for var i := N2 downto N1 do
begin
  цела цыкла;
end;
```

Прыклад 6.3. Агульны выгляд працэдуры:

```
procedure <імя>(<спіс параметраў>:тып);
var <апісанне пераменных>
begin
  <каманды>
end;
```

Агульны выгляд функцыі:

```
function  <імя>(<спіс параметраў>:тып): тып выніку;
var <апісанне пераменных>
begin
  <каманды>
  <імя> := <значэнне>;
end;
```

Функцыя павінна змяшчаць каманду выгляду <імя> := <значэнне>;. Гэта каманда вызначае, што функцыя павінна вярнуць у якасці выніку.

Прыклад 6.4. Выклік працэдуры і функцыі.

Выклік працэдуры малявання круга:

```
Circle(250, 125, 30);
```

выклік функцый для вылічэння квадратнага кораня і сінуса:

```
d := sqrt(2) * sin(x);
```

Прыклад 6.5. Цэлалікавыя тыпы даных у PascalABC.

Тып	Дыяпазон значэнняў	Памер памяці, байт
shortint	−128..127	1
smallint	−32768..32767	2
integer, longint	−2147483648.. 2147483647	4
byte	0..255	1
word	0..65535	2
longword, cardinal	0..4294967295	4

Дадаткова ў PascalABC вызначаны тып BigInteger, які не абмежаваны дыяпазонам значэнняў і памерам памяці.

Прыклад 6.6. Рэчыўныя тыпы даных у PascalABC.

Тып	Дыяпазон значэнняў	Памер памяці, байт
real (double)	$-1.8\cdot10^{308}..1.8\cdot10^{308}$	8
single	$-3.4\cdot10^{38}..3.4\cdot10^{38}$	4
decimal	$-(2^{96}-1)..2^{96}-1$	16

Колькасць значных лічбаў у тыпе real складае 15—16, у тыпе single — 7—8, у тыпе decimal — 28—29.

Тып real мае іншую назву — double. Самы маленькі дадатны лік тыпу real прыблізна роўны $5.0\cdot10^{-324}$, для тыпу single ён складае прыблізна $1.4\cdot10^{-45}$.

выкарыстоўвацца толькі ўнутры дадзенай працэдуры або функцыі.

Функцыі, у адрозненне ад працэдур, у выніку свайго выканання вяртаюць значэнне, якое можа быць выкарыстана ў выразе. Выклік працэдуры з'яўляецца асобнай камандай (прыклад 6.4).

6.3. Тыпы даных

У мове Pascal выкарыстоўваюцца розныя тыпы даных. Яны патрэбныя для выканання розных аперацый — з кожным тыпам даных звязаны свой набор аперацый.

Для захоўвання розных тыпаў даных у памяці камп'ютара адводзіцца розная колькасць памяці. Вы ўжо выкарыстоўвалі тыпы даных, якія называюць простымі. У табліцы прыкладу 6.5 прыведзены цэлалікавыя тыпы даных, якія выкарыстоўваюцца ў PascalABC.

Усе цэлалікавыя тыпы даных, прыведзеныя ў табліцы, можна падзяліць на дзве групы:

- **знакавыя** (дыяпазон значэнняў якіх змяшчае як дадатныя, так і адмоўныя лікі);
- **бяззнакавыя** (дыяпазон значэнняў змяшчае толькі неадмоўныя лікі).

Для кожнага знакавага тыпу ёсць бяззнакавы, які займае столькі ж памяці.

Рэчыўныя тыпы даных дазваляюць захоўваць лік, прыведзены ў стандартным выглядзе: $a\cdot10^n$, дзе $1\leqslant a<10$

і *n* (цэлае) ёсць парадак ліку, запісанага ў стандартным выглядзе (прыклад 6.6).

Значэнні тыпу boolean, які называюць лагічным, займаюць 1 байт і прымаюць адно з двух значэнняў, зададзеных канстантамі true (праўда) і false (няпраўда).

Даныя ў праграму карыстальнік можа ўводзіць з дапамогай каманды read (readln). Для вываду даных выкарыстоўваецца каманда write (writeln).

Аператар прысвойвання выкарыстоўваецца для таго, каб задаваць значэнні пераменным і вылічваць значэнне выразу.

Пры выкарыстанні ў адным аператары прысвойвання даных розных тыпаў трэба памятаць пра іх сумяшчальнасць:

- пераменнай цэлалікавага тыпу нельга прысвоіць рэчыўнае значэнне;
- для даных рэчыўных тыпаў вызначаны аперацыі «+», «−», «*», «/»;
- для даных цэлалікавых тыпаў вызначаны аперацыі «+», «−», «*», «**div**», «**mod**».

Цэлалікавыя тыпы могуць быць ператвораны ў рэчыўныя, але не наадварот (прыклад 6.7).

Усе простыя тыпы, акрамя рэчыўнага, называюцца **парадкавымі**. Значэнні толькі гэтых тыпаў могуць быць параметрамі цыкла **for**. Для парадкавых тыпаў выкарыстоўваюцца функцыі ord, pred і succ, а таксама працэдуры inc і dec (прыклад 6.8

Прыклад 6.7. Прывядзенне тыпаў.

```
var a, b, c: real;
    x, y, z:  integer;
begin
  a := 1; b := 3; x := 6; y := 4;
  //вылічэнні з рэчыўным тыпам
  c := a / b; writeln(c);
  //пераўтварэнне  да рэчыўнага
  c := x / y; writeln(c);
  //вылічэнні з цэлым тыпам
  z := x div y; writeln(z);
  //пераўтварэнне да рэчыўнага
  c := x div y; writeln(c);
end.
```

Вынік:

```
Окно вывода
0.333333333333333
1.5
1
1
```

Прыклад 6.8. Працэдуры і функцыі для работы з парадкавымі тыпамі.

Функцыя	Апісанне
Ord(a)	Парадкавы нумар значэння a
Pred(x)	Значэнне, якое папярэднічае x
Succ(x)	Значэнне, наступнае за x

Працэдура	Апісанне
Inc(i);	Павялічвае значэнне пераменнай i на 1
Inc(i, n)	Павялічвае значэнне пераменнай i на n
Dec(i);	Памяншае значэнне пераменнай i на 1
Dec(i, n)	Памяншае значэнне пераменнай i на n

Прыклад 6.9. Фрагмент праграмы для работы з парадкавымі тыпамі.

```
var a, c: integer;
    b: boolean;
begin
  a := 10; c := 3; b := true;
  writeln(ord(a));
  writeln(pred(b));
  writeln(succ(c));
  inc(b); writeln(b); dec(c, a);
  writeln(c);
end.
```

```
Окно вывода
10
False
4
False
-7
```

Прыклад 6.10.

V. Праграма:

```
uses GraphABC;
const slovo = 'Pascal';
begin
  SetFontColor(clRed);
  writeln(slovo);
end.
```

VI. Тэсціраванне.

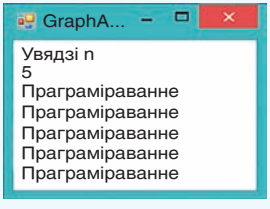
Запусціць праграму. Вынік:

Прыклад 6.11.

V. Праграма:

```
uses GraphABC;
const sl ='Праграміраванне';
var n: integer;
begin
  writeln ('Увядзі n ');
  read(n); writeln (n);
  for var i:= 1 to n do
    writeln(sl);
end.
```

VI. Тэсціраванне. Вынік пры $n = 5$:



і прыклад 6.9). Усе пераменныя, якія выкарыстоўваюцца ў праграме, павінны быць апісаны ў частцы **var**. Калі даныя не змяняюцца ў працэсе работы праграмы, то яны могуць быць апісаны як канстанты ў частцы **const**. Напрыклад:

```
const slovo = 'Прывітанне';
      Pi = 3.1416;
```

Для работы з графічнымі данымі выкарыстоўваюцца каманды бібліятэкі GraphABC (гл. *Дадатак 2*, с. 159—160).

6.4. Прыклады задач

Прыклад 6.10. Апісаць слова «Pascal» як канстанту. Вывесці слова на экран чырвоным колерам.

Этапы выканання задання

I—II. Вынік работы не залежыць ад зыходных даных.

III. Алгарытм рашэння задачы.

1. Устанавіць чырвоны колер (каманда знаходзіцца ў бібліятэцы GraphABC).

2. Вывесці канстанту.

IV. У праграме няма пераменных.

Прыклад 6.11. Напісаць праграму, якая выведзе зададзенае слова на экран n разоў. Значэнне n уводзіцца.

Этапы выканання задання

I. Зыходныя даныя: пераменная n.

II. Вынік: n слоў.

III. Алгарытм рашэння задачы.

1. Апісваем слова як канстанту.

2. Уводзім значэнне n.

3. Для вываду слова n разоў выкарыстаем цыкл **for**.

4. Выводзім слова ў цыкле.

IV. Апісанне пераменных: n — integer.

Прыклад 6.12. Напісаць праграму, якая выведзе на экран n разоў адно з двух слоў. Выбар слова ажыццяўляецца выпадковым чынам. Значэнне n уводзіцца. Палічыце, колькі разоў было выведзена кожнае слова.

Этапы выканання задання

I. Зыходныя даныя: пераменная n.

II. Вынік: n разоў выведзена адно з двух слоў і паведамленне пра тое, колькі разоў выведзена кожнае са слоў.

III. Алгарытм рашэння задачы.

1. Словы апісваем як канстанты.

2. Уводзім значэнні n.

3. Ініцыялізуем нулём пераменныя k1 і k2, якія будуць падлічваць, колькі разоў выведзена кожнае слова.

4. Для вываду слоў выкарыстаем цыкл **for**.

4.1. Згенеруем выпадковы лік x на прамежку [0; 2).

4.2. Калі x = 0, то выведзем першае слова і павялічым значэнне пераменнай k1 на 1.

4.3. Інакш выведзем другое слова і павялічым значэнне пераменнай k2 на 1.

5. Выводзім паведамленні пра колькасць слоў.

IV. Апісанне пераменных: n, k1, k2, x — integer.

Прыклад 6.12.

V. Праграма:

```
uses GraphABC;
const sl1 = 'Pascal';
      sl2 = 'Python';
var n, k1, k2, x: integer;
begin
 writeln ('Увядзі  n ');
 read(n);
 writeln (n);
 k1 := 0; k2 := 0;
 for var i:=1 to n do
 begin
  x := random(2);
  if x = 0 then
  begin
   writeln(sl1);
   k1 := k1 + 1;
  end
  else
  begin
   writeln(sl2);
   k2 := k2 + 1;
  end
 end;
 writeln('Слова ',sl1,' - ',k1);
 writeln('Слова ',sl2,' - ',k2);
end.
```

VI. Тэсціраванне. Запусціць праграму. Вынік пры $n = 8$:

?

1. Што такое алгарытм?
2. Назавіце асноўныя алгарытмічныя канструкцыі.
3. Што разумеюць пад дапаможным алгарытмам?
4. Чым адрозніваюцца розныя цэлалікавыя тыпы даных адзін ад аднаго?
5. Што трэба памятаць пра сумяшчальнасць тыпаў даных?
6. Якія арыфметычныя аперацыі вызначаны для цэлалікавых тыпаў даных? Для рэчыўных тыпаў даных?

Практыкаванні

1 Змяніце канстанту ў праграме з прыкладу 6.10 на сваё імя. Выкарыстоўваючы каманды графічнай бібліятэкі для работы з тэкстам, змяніце шрыфт, памер сімвалаў, фон, напісанне літар.

2 Змяніце праграму прыкладу 6.11 так, каб выконваліся дадзеныя ўмовы.

1. Кожнае слова павінна выводзіцца выпадковым колерам.
2*. Адлегласць паміж словамі павінна быць 50 пікселяў.
3*. Кожнае новае слова павінна выводзіцца шрыфтам, на 5 пунктаў большым, чым папярэдняе. Адрэгулюйце адлегласць паміж словамі так, каб словы пры вывадзе не перакрывалі адно другое.

3 Запусціце праграму з прыкладу 6.12 некалькі разоў. Якія вынікі атрымалі?

4 Змяніце праграму з прыкладу 6.12 так, каб выпадковым чынам выбіралася адно з трох слоў. Выводзьце кожнае слова сваім колерам (напрыклад, першае — чырвоным, другое — сінім, трэцяе — зялёным).

§ 7. Радковыя велічыні

У першых мовах праграмірования радковага тыпу даных не было; праграміст павінен быў сам будаваць функцыі для работы з радкамі.

У 1962 г. была распрацавана мова SNOBOL (StriNg Oriented symBOlic Language), арыентаваная на работу з радкамі. У канцы 60-х гг. XX ст. радковыя тыпы даных з'явіліся ў мовах Algol і Fortran.

Два радкі, у адрозненне ад двух лікаў, нельга прачытаць з дапамогай адной каманды `read`, паколькі прабел для радкоў не раздзяляльнік, а такі ж сімвал, як і ўсе астатнія. Неабходна выкарыстоўваць дзве каманды `readln`.

Калі выкарыстоўваць дзве каманды `read`, то першы радок будзе счытаны так, як трэба, а другі радок будзе пусты (ён не будзе ўводзіцца). Гэта адбываецца таму, што першая каманда `read` счытвае даныя да націску клавішы Enter. Другая каманда `read` прачытае адзін сімвал — сімвал націскання клавішы Enter.

7.1. Увод, вывад, прысвойванне радковых велічынь

Сучасныя камп'ютары здольныя апрацоўваць даныя, прыведзеныя рознымі спосабамі: лікі, тэксты, графіку, гукі. Вы ўжо ведаеце, як на мове праграмірования Pascal можна працаваць з цэлымі і рэчаіснымі лікамі, выконваць найпрасцейшыя графічныя пабудовы. Апрацоўка тэкставых даных з'яўляецца сённяння найбольш актуальнай — гэта апрацоўка розных пошукавых запытаў у Інтэрнэце, пераклад тэкстаў з адной мовы на іншую, агучванне камп'ютарам друкаванага тэксту і інш.

У мове Pascal для работы з тэкставымі данымі выкарыстоўваецца тып **`string` (радок).** Радкі складаюцца з набору паслядоўна размешчаных сімвалаў і выкарыстоўваюцца для захоўвання тэксту. Яны могуць мець ад-