§ 14. Мова праграмірavання Паскаль

Камп'ютар (ад англ. *computer* — вылічальнік) — устройства або сістэма, здольныя выконваць зададзеную дакладна пэўную змяняемую паслядоўнасць аперацый (часцей за ўсё лікавых разлікаў).

Электронна-вылічальная машына (ЭВМ) — комплекс тэхнічных сродкаў, дзе асноўныя функцыянальныя элементы (лагічныя; якія запамінаюць; індыкацыйныя і інш.) выкананы на электронных прыборах, прызначаных для аўтаматычнай апрацоўкі інфармацыі ў працэсе рашэння вылічальных задач.

Ніклаус Вірт (нарадзіўся ў 1934 г.) — швейцарскі вучоны, спецыяліст па інфарматыцы, адзін з самых вядомых тэарэтыкаў у галіне распрацоўкі моў праграмірavання, прафесар камп'ютарных навук. Стваральнік і вядучы праекціроўшчык моў праграмірavання Паскаль, Модула-2, Аберон.

Жаданне спрасціць і паскорыць разнастайныя разлікі ўласціва чалавеку са старажытных часоў. Сёння камп'ютар здольны выконваць сотні мільёнаў аперацый у секунду. Для рашэння вылічальных задач патрабуецца спачатку скласці алгарытм іх рашэння, а затым запісаць яго ў выглядзе праграмы, выкарыстоўваючы якую-небудзь мову праграмірavання.

Мова праграмірavання вызначае набор правіл, якія акрэсліваюць знешні выгляд праграмы і дзеянні, што ажыццявіць выканаўца пад яе кіраваннем.

Мова праграмірavання Паскаль (Pascal) выкарыстоўваецца для навучання праграмірavанню і з'яўляецца базай для шэрага прафесійных моў праграмірavання.

Існуе мноства асяроддзяў праграмірavання, якія падтрымліваюць мову Паскаль: PascalABC, FreePascal, Delphi, GNU Pascal, Dev-Pascal, Rad Studio і інш. У вучэбным курсе выкарыстоўваецца асяроддзе PascalABC (з ім вы працавалі, знаёмячыся з вучэбнымі камп'ютарнымі выканаўцамі).

14.1. Каманда вываду

Дэманстраваць работу любой праграмы мае сэнс толькі тады, калі яна выводзіць якую-небудзь інфармацыю.

Праграма на мове Pascal (цела праграмы) павінна пачынацца са слова **begin**, а заканчвацца словам **end** і кропкай. Праграма, што складаецца з гэтых каманд, раздзеленых прабелам або пераводам радка, можа быць запушчана на выкананне, але яна нічога не робіць. Дабавім у яе каманду вываду прывітання:

```
begin
  write('Прывітанне!');
end.
```

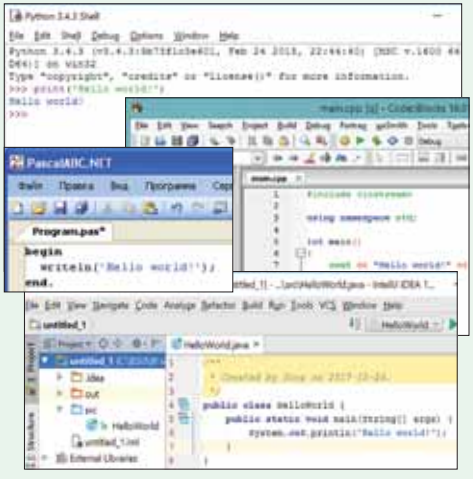
Вынік работы праграмы адлюстроўваецца ў ніжняй частцы акна праграмы PascalABC у акне вываду (прыклад 14.1).

Каманда `write( );` прызначана для **вываду даных.**

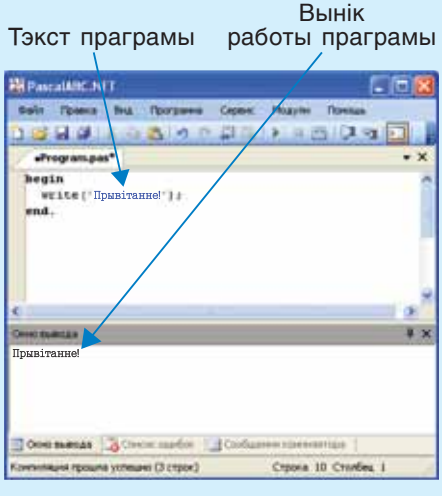
Тэкст, які трэба вывесці на экран, змяшчаюць у апострафах (адзінарнае двукоссе). Гэты тэкст не аналізуецца і выводзіцца ў тым выглядзе, у якім ён запісаны. Тэкст можна запісаць на любой мове. Тэкстам можа быць адвольны набор сімвалаў.

У адной праграме можа быць некалькі каманд вываду. Для

Па традыцыі, якая пачалася ў 1978 г. з прыкладу ў кнізе Б. Кернігана і Д. Рытчы «Мова праграміравання Сі», першая праграма на любой мове праграміравання павінна выводзіць на экран прывітанне свету:



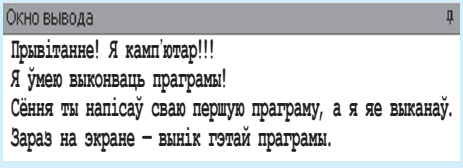
Прыклад 14.1. Акно асяроддзя PascalABC з вынікам работы праграмы:



Прыклад 14.2. Тэкст праграмы:

```
begin
  write('Прывітанне!  ');
  writeln('Я  камп'ютар!!!');
  write('Я ўмею выконваць ');
  writeln('праграмы!');
  write('Сёння  ты  ');
  write('напісаў  сваю  ');
  write('першую праграму,');
  writeln(' а я яе выканаў.');
  write('Зараз  на  экране  –');
  writeln('  вынік  гэтай
праграмы.');
end.
```

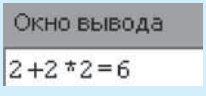
Вынік работы праграмы:



Прыклад 14.3. Тэкст праграмы:

```
begin
  write('2+2*2=');
  write(2+2*2);
end.
```

Вынік работы праграмы:



Дзве каманды `write` ў праграме можна аб'яднаць у адну, аддзяліўшы тэкст ад выразу коскай:

```
begin
  write('2+2*2=',  2+2*2);
end.
```

выводу тэксту, запісанага ў некалькі радкоў, выкарыстоўваюць каманду `writeln( )`. Спалучэнне «`ln`» (скар. ад англ. *line* — лінія, радок), запісанае ў канцы каманды, азначае, што пасля выводу трэба перавесці курсор у новы радок.

Прыклад 14.2. Выведзем на экран камп'ютара наступны тэкст: «`Прывітанне! Я камп'ютар!!! Я ўмею выконваць праграмы! Сёння ты напісаў сваю першую праграму, а я яе выканаў. Зараз на экране – вынік гэтай праграмы`».

Выкарыстоўваючы спалучэнне каманд `write` і `writeln`, тэкст можна размясціць па-рознаму.

Як вы ўжо ведаеце, тэкст у камандзе `write( )`, запісаны ў двукоссі, не аналізуецца. Калі двукоссе прапусціць, то выконваецца аналіз даных, запісаных у дужках. Так, калі ў дужках напісаць арыфметычны выраз, то спачатку вылічваецца яго значэнне, а затым выводзіцца вынік.

Прыклад 14.3. Вылічым значэнне выразу 2 + 2 * 2.

Калі запісаць выраз у двукоссі, то будзе выведзены сам выраз. Пры адсутнасці двукосся на экран будзе выведзена значэнне дадзенага выразу.

14.2. Паняцце тыпу даных

На практыцы рэдка даводзіцца пісаць праграмы, якія рашаюць толькі адну задачу. Звычайна праграмы пішуцца для рашэння цэлага класа задач, што можна сфармуляваць у агульным выглядзе.

З такімі задачамі вы ўжо сустракаліся ў курсе матэматыкі. Напрыклад, рашэнне задачы «Знайдзіце плошчу прамавугольніка» можна запісаць так: $S = a \cdot b$, дзе пераменныя a і b абазначаюць адпаведна даўжыню і шырыню прамавугольніка, а S — плошчу. Ведаючы гэту формулу, можна знайсці плошчу любога прамавугольніка.

У праграмiраванні для рашэння задач у агульным выглядзе таксама выкарыстоўваюць пераменныя. Паколькі з такімі пераменнымі будзе працаваць камп'ютар, то яны павінны захоўвацца ў яго памяці.

Інфармацыю, пададзеную ў прыдатным для апрацоўкі на камп'ютары выглядзе, называюць **данымі**.

Пераменная ў праграмiраванні — гэта найменная ячэйка памяці, якая захоўвае значэнне пераменнай.

Да пачатку 1950-х гг. XX ст. праграмісты ЭВМ пры стварэнні праграм карысталіся машынным кодам. Запіс праграмы на машынным кодзе складаўся з адзінак і нулёў. Машынны код прынята лічыць мовай праграмiравання першага пакалення. Тыпы даных не выкарыстоўваліся.

Першай мовай праграмiравання, у якой з'явілася магчымасць ствараць пераменныя, лічыцца Асэмблер. У гэтай мове замест машынных кодаў сталі выкарыстоўваць каманды, запісаныя тэкстам. Асэмблер належыць да моў праграмiравання другога пакалення.

У 1957 г. з'явілася мова Фартран, якая адкрыла эру моў праграмiравання трэцяга пакалення. Яна дазволіла выкарыстоўваць розныя лікавыя тыпы даных, неабходныя для складаных разлікаў: цэлыя, рэчыўныя (рэчаісныя) і комплексныя.

Далейшае развіццё моў праграмiравання дазволіла дабавіць магчымасць работы з іншымі тыпамі даных. Сучасныя мовы праграмiравання дазваляюць працаваць з вялікай колькасцю тыпаў даных.

У асяроддзі праграміравання PascalABC рэалізавана больш за 30 розных тыпаў даных.

Обзор типов

Типы в PascalABC.NET подразделяются на простые, структурированные, типы указателей, процедурные типы, последовательности и классы.

К *простым* относятся целые и вещественные типы, логический, символьный, перечислимый и диапазонный тип.

Тип данных называется *структурированным*, если в одной переменной этого типа может содержаться множество значений.

К структурированным типам относятся массивы, строки, записи, кортежи, множества, файлы и классы.

Особым типом данных является последовательность, которая хранит по-существу алгоритм получения данных последовательности один за другим.

Все простые типы, кроме вещественного, называются *порядковыми*. Только значения этих типов могут быть индексами статических массивов и параметрами цикла `for`. Кроме того, для порядковых типов используются функции `Ord`, `Pred` и `Succ`, а также процедуры `Inc` и `Dec`.

Прыклад 14.4. Прыклады апісання пераменных:

```
var x: real;
var x1, y1: real;
var a_1, a_2, a_3: real;
```

Дыяпазон магчымых значэнняў тыпу `real` задаецца лікамі ў стандартным уяўленні ад $-1.8 \cdot 10^{308}$ да $1.8 \cdot 10^{308}$. Найменшы дадатны лік тыпу `real` прыбліжана роўны $5.0 \cdot 10^{-324}$. Пры вылічэннях у ліку захоўваецца да 16 лічбаў.

Камп'ютар можа апрацоўваць даныя розных тыпаў: цэлыя і рэчаісныя лікі, сімвалы, тэксты і інш.

Тып даных вызначае спосаб захоўвання даных у памяці камп'ютара, дыяпазон магчымых значэнняў даных і аперацыі, якія з гэтым тыпам даных можна выконваць.

Каб выкарыстаць якую-небудзь пераменную, яе трэба апісаць. Апісанне пераменных выконваецца да пачатку праграмы (каманды **begin**) (прыклад 14.4). Пры апісанні пераменнай вылучаецца памяць для захоўвання яе значэння. У працэсе выканання праграмы значэнне пераменнай можа змяняцца.

Для апісання пераменных выкарыстоўваецца каманда **var** (скар. ад англ. *variable* — пераменная). Фармат запісу каманды:

var <імя пераменнай>: <тып>;

Для абазначэння імя пераменнай выкарыстоўваюць літары лацінскага алфавіта, лічбы і знак «_». Першым знакам павінна быць літара або знак падкрэслівання.

Тып даных `real` у мове Pascal дазваляе працаваць з лікамі і выконваць над імі арыфметычныя дзеянні.

14.3. Аператар прысвойвання

Адной з асноўных каманд для апрацоўкі даных у праграме з'яўляецца аператар прысвойвання.

Аператар прысвойвання прызначаны для таго, каб:

- задаваць значэнні пераменным;
- вылічваць значэнні арыфметычнага выразу (вынік вылічэння будзе запісаны як значэнне пераменнай).

Фармат запісу аператара:

<ім'я пераменнай>:= <выраз>;

(Разгледзьце прыклад 14.5.)

У запісе арыфметычнага выразу выкарыстоўваюцца знакі матэматычных дзеянняў.

Матэматычныя аперацыі	Запіс у Pascal
+ (складанне)	+
− (адніманне)	-
· (множанне)	*
: (дзяленне)	/

Прыярытэт выканання аперацый адпавядае прынятаму ў матэматыцы: спачатку выконваюцца множанне і дзяленне, а затым складанне і адніманне. Для змянення парадку дзеянняў у выразах выкарыстоўваюць дужкі.

Прыклад 14.5. Прыклады запісу аператара прысвойвання:

```
x:= 7;
x1:= 3.5;
a_1: = 20 * (x + x1) - 32;
y:= y + 7;
```

Прыклад 14.6. Запішам аператар прысвойвання на Pascal для матэматычных выразаў:

Выраз	Запіс на Pascal
$S = 2(a + b)$	`S:=2*(a+b);`
$S = a^2$	`S:=a*a;`
$a = \frac{x + y}{3}$	`a:=(x+y)/3;`

Прыклад 14.7. Запішам аператар прысвойвання, пасля выканання якога значэнне пераменнай a павялічыцца ў 2 разы, а пераменнай b паменшыцца на 3.

У Pascal дапушчальныя каманды прысвойвання наступнага выгляду:

```
a:= a * 2;
```

Сэнс такой каманды наступны: з ячэйкі памяці здабываецца значэнне пераменнай a, затым яно памнажаецца на 2, вынік запісваецца ў тую ж ячэйку памяці. Старое значэнне пераменнай a будзе страчана.

Запіс аператара прысвойвання для змянення значэння пераменнай b наступны:

```
b:= b - 3;
```

У PascalABC.NET вызначаны аператары прысвойвання са значкамі +=, −=, *=, /=. Яны дазваляюць змяніць значэнне пераменнай. Напрыклад:

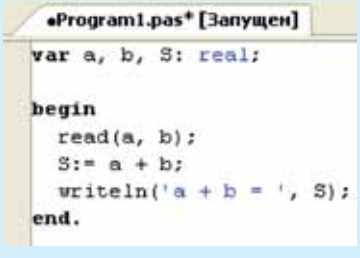
```
a *= 2;  // павялічыць a ў 2 разы;
b -= 3;  // паменшыць b на 3.
```

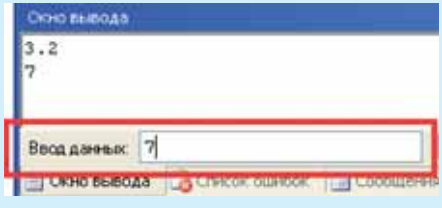
Прыклад 14.8. Увесці два лікі, знайсці і вывесці іх суму.

Тэкст праграмы:

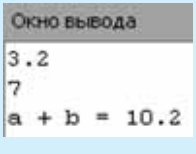
```
var a, b, S: real;
begin
 read(a, b);
 S:= a + b;
 writeln('a + b =', S);
end.
```

Увод даных:





Вынік:



Для запісу звычайнага дробу выкарыстоўваецца знак дзялення. Знак множання апускаць нельга. Цэлая частка дробавага ліку аддзяляецца ад дробавай часткі кропкай. (Разгледзьце прыклады 14.6 і 14.7 на с. 93.)

14.4. Увод даных

Пачатковыя значэнні пераменным можна задаваць не толькі з дапамогай аператара прысвойвання, але і шляхам уводу з клавіятуры. У гэтым выпадку, калі неабходны вылічэнні з новым наборам значэнняў зыходных даных, тэкст праграмы не трэба змяняць.

Каманда `read( )` прызначана для **ўводу даных.** У дужках праз коску пералічаюцца імёны пераменных, значэнні якіх неабходна ўвесці.

Увод даных адбываецца ў ніжняй частцы акна праграмы PascalABC. Для гэтага выкарыстоўваецца акно «Увод даных». Пасля націскання на кнопку «Увесці» або на клавішу «Enter» уведзеныя значэнні пераносяцца ў акно выводу. Пасля завяршэння работы праграмы ў гэтым жа акне будзе выведзены вынік (прыклад 14.8).

14.5. Структура праграмы

Усе праграмы на мове праграміравання Pascal маюць агульную структуру. У праграме можна вылучыць наступныя часткі:

- загаловак (неабавязковы);
- бібліятэкі, што падключаюцца (модулі) (калі падключаць дадатковыя бібліятэкі не трэба, раздзел адсутнічае; вядомыя бібліятэкі: `Drawman`, `Robot`, `RobTasks`);
- апісанне пераменных з вызначэннем іх тыпу;
- апісанне дапаможных алгарытмаў (калі выкарыстоўваць дапаможныя алгарытмы не трэба, раздзел адсутнічае);
- **begin ... end.** — службовыя словы, што абрамляюць цела асноўнай праграмы, у якой знаходзяцца выконваемыя каманды; **begin** пачынае выконваемую частку праграмы, а **end.** (кропка ў канцы абавязковая) яе завяршае.

У мінімальна магчымым наборы праграма складаецца з пустога цела праграмы: **begin end.** Праграма, што змяшчае ўсе раздзелы, паказана ў прыкладзе 14.9.

Для кожнага раздзела вызначана ключавое службовае слова, якім ён пачынаецца. Пры напісанні праграмы ключавыя словы вылучаюцца паўтлустым шрыфтам.

Прыклад 14.9. Праграма, якая змяшчае ўсе раздзелы (падлічваецца колькасць зафарбаваных клетак у полі Робата памерам 10 × 10):

```
//загаловак праграмы
//(неабавязкова)
program Primer;
//апісанне бібліятэк
uses Robot, RobTasks;
//апісанне пераменных
var k: integer;
//апісанне дапаможных
//алгарытмаў
procedure line;
begin
 for var i:= 1 to 9 do
 begin
  if CellIsPainted then
   k:= k + 1;
  right;
 end;
 if CellIsPainted then
  k:= k + 1;
end;
procedure back;
begin
 for var i:= 1 to 9 do
  left;
end;
//асноўная праграма
begin
//цела праграмы
 Task('myrob11');
 k:= 0;
 for var i:= 1 to 9 do
 begin
  line; back; down;
 end;
 line;
 writeln(k);
end.
```

1. Якая каманда мовы праграміравання Pascal прызначана для выведу даных?
2. Што вызначае тып даных?
3. Для чаго выкарыстоўваецца каманда прысвойвання?
4. Якая каманда мовы праграміравання Pascal прызначана для ўводу даных?
5. З якіх раздзелаў складаецца праграма на мове праграміравання Pascal?

Практыкаванні

1 Для праграмы з прыкладу 14.2 выканайце наступныя заданні (файл з праграмай можна спампаваць):

1. Замяніце ўсе каманды `writeln` на каманды `write` і выканайце праграму. Што адбылося? Растлумачце чаму.
2. Як зменіцца вынік работы праграмы, калі ў зыходным тэксце замяніць усе каманды `write` на `writeln`?
3. Змяніце праграму так, каб тэкст на экране выглядаў наступным чынам:

```
Прывітанне! Я камп'ютар!!! Я ўмею выконваць праграмы!
Ты сёння напісаў сваю першую праграму!!!
Я выканаў тваю праграму. Паглядзі на экране вынік!
```

2 Унясіце неабходныя змяненні ў праграму з прыкладу 14.3, каб дзеянні выконваліся ў тым парадку, у якім запісаны, г. зн. спачатку складанне, а потым множанне.

3 Уводзіцца ўзрост карыстальніка ў гадах. Вызначыце ўзрост карыстальніка праз 5 гадоў.

4 Напішыце праграму, у якой уводзяцца два лікі *a* і *b*. Затым першы лік памяншаецца ў 2 разы, а другі павялічваецца на 30. Выведзіце зменены значэнні пераменных.

5 Напішыце праграму для вылічэння значэння лікавых выразаў:

1. $23 + 45 \cdot 11 - 15$.
2. $\dfrac{37 + 2 \cdot 27}{41}$.
3. $\dfrac{5638 - 2347}{49} + \dfrac{123 \cdot 756}{4455}$.

§ 15. Арганізацыя вылічэнняў

Пры рашэнні любой задачы чалавеку даводзіцца выконваць наступныя дзеянні:

- вызначэнне зыходных даных (што дадзена ў задачы);
- вызначэнне вынікаў (што трэба атрымаць);
- апрацоўка зыходных даных у адпаведнасці з вядомымі правіламі так, каб атрымаць вынік.

Ужываючы дадзеныя правілы пры рашэнні задачы па праграмiраванню, атрымаем наступныя этапы рашэння задачы:

I. Вызначэнне зыходных даных.

II. Вызначэнне вынікаў.

III. Складанне алгарытму рашэння задачы.

IV. Вызначэнне тыпаў даных для пераменных, што выкарыстоўваюцца пры рэалізацыі алгарытму.

V. Напісанне праграмы.

VI. Тэсціраванне праграмы.

VII. Аналіз вынікаў.

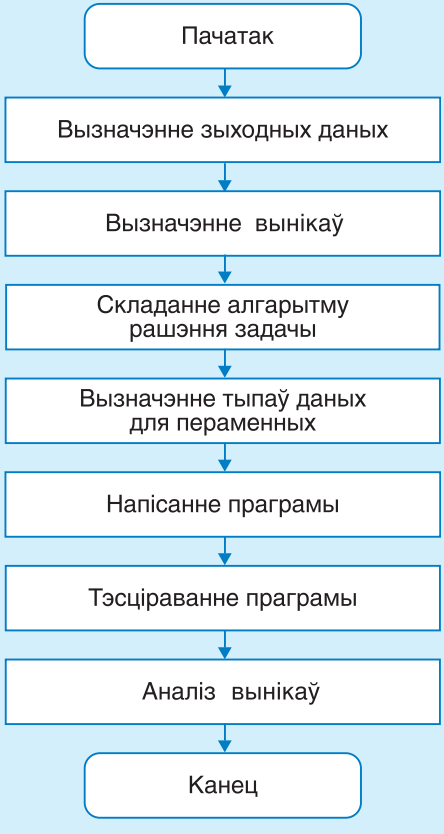
(Разгледзьце прыклад 15.1.)

Тэсціраванне праграмы — праверка правільнасці работы праграмы пры розных наборах зыходных даных.

Прыклад 15.1. Рашэнне задач па фізіцы прынята афармляць пэўным чынам.

Злева запісваецца тое, што дадзена і што трэба атрымаць, справа — паслядоўнасць дзеянняў, якая прыводзіць да рашэння задачы. Аналагічна афармляюцца рашэнні задач па хіміі, геаметрыі.

Этапы рашэння задачы па праграмiраванні можна паказаць наступным чынам:



Прыклад 15.2.

V. Праграма:

```
var  x, y, z, a:  real;
begin
 write('увядзіце  x  =  ');
 read(x);
 write('увядзіце  y  =  ');
 read(y);
 write('увядзіце  z  =  ');
 read(z);
 a:=(2*x+3*y-z)/(3+x*x);
 writeln('a  =  ',a);
end.
```

VI. Тэсціраванне праграмы.

Запусціце праграму і ўвядзіце значэнні: $x = 2$, $y = 3$, $z = 1$.

Вынік работы праграмы павінен быць наступным:

```
Окно вывода
увядзіце x = 2
увядзіце y = 3
увядзіце z = 1
a = 1.71428571428571
```

А для значэнняў $x = 2$, $y = 4$, $z = 2$ атрымаем:

```
Окно вывода
увядзіце x = 2
увядзіце y = 4
увядзіце z = 2
a = 2
```

VII. Праверка правільнасці вылічэнняў можа быць выканана на калькулятары.

15.1. Вылічэнне значэння арыфметычнага выразу

Прыклад 15.2. Дадзены пераменныя x, y, z. Напішам праграму для вылічэння значэння выразу $a = \dfrac{2x + 3y - z}{3 + x^2}$.

Этапы выканання задання:

I. Вызначэнне зыходных даных: пераменныя x, y, z.

II. Вызначэнне вынікаў: пераменная a.

III. Алгарытм рашэння задачы:

1. Увод зыходных даных.
2. Вылічэнне значэння выразу.
3. Вывад выніку.

IV. Апісанне пераменных.

Усе пераменныя, вызначаныя для рашэння задачы, маюць тып `real`.

У прыведзеным прыкладзе перад кожнай камандай уводу запісана каманда выводу з тлумачэннямі пра тое, значэнне якой пераменнай трэба ўводзіць.

Пры напісанні праграм для вылічэння значэння арыфметычнага выразу часта дапускаюць наступныя памылкі:

`2x+3;` `(x-4)(x+2);`	Прапушчаны знак *
`(2+y)/(x*x;`	Прапушчана дужка

Будзьце ўважлівымі!

15.2. Выкарыстанне мовы праграмiравання для рашэння задач

Прыклад 15.3. Напішам праграму для рашэння геаметрычнай задачы. Зададзены квадрат з даўжынёй стараны a. Патрабуецца знайсці яго плошчу і перыметр.

Этапы выканання задання:

I. Вызначэнне зыходных даных: пераменная a (даўжыня стараны).

II. Вызначэнне вынікаў: пераменныя S (плошча) і P (перыметр).

III. Алгарытм рашэння задачы:

1. Увод зыходных даных.

2. Вылічэнне значэнняў плошчы ў матэматыцы выконваецца па формуле $S = a^2$, а перыметра — па формуле $P = 4a$. У праграме гэтым формулам будуць адпавядаць каманды прысвойвання:

$S := a * a$; $P := 4 * a$.

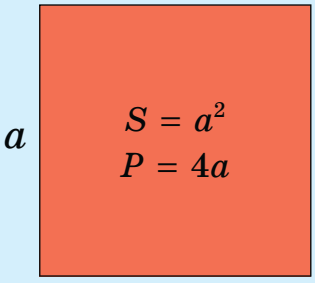
3. Вывад выніку.

IV. Апісанне пераменных:

Усе пераменныя, вызначаныя для рашэння задачы, маюць тып `real`.

Звярніце ўвагу: запіс формул у аператары прысвойвання можа адрозніваццa ад запісу матэматычных формул.

Прыклад 15.3.



V. Праграма:

```
var a,S,P: real;
begin
 write('увядзіце a = ');
 read(a);
 s:= a*a;
 p:= 4*a;
 writeln('плошча = ',S);
 writeln('перыметр = ',P);
end.
```

VI. Тэсціраванне праграмы.

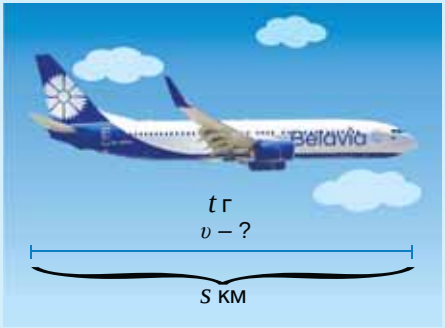
Запусціце праграму і ўвядзіце значэнне $a = 5.2$.

Вынік работы праграмы павінен быць наступным:

```
Окно вывода
увядзіце a = 5.2
плошча = 27.04
перыметр = 20.8
```

VII. Праверка правільнасці вылічэнняў можа быць выканана на калькулятары.

Прыклад 15.4.



V. Праграма:

```
var s, t, v: real;
begin
 write('увядзіце s = ');
 read(s);
 write('увядзіце t = ');
 read(t);
 v:= s / t;
 writeln('скорасць = ', v);
end.
```

VI. Тэсціраванне праграмы.

Запусціце праграму і ўвядзіце значэнні $s = 3550$ і $t = 4$.

Вынік работы праграмы павінен быць наступным:

```
Окно вывода
увядзіце s = 3550
увядзіце t = 4
скорасць = 887.5
```

VII. Праверка правільнасці вылічэнняў можа быць выканана на калькулятары.

Прыклад 15.4. Напішам праграму для рашэння фізічнай задачы. Адлегласць паміж двума гарадамі складае s км. Самалёт пралятае гэту адлегласць за t г. Вызначыце скорасць самалёта.

Этапы выканання задання:

I. Вызначэнне зыходных даных: пераменныя s (адлегласць) і t (час).

II. Вызначэнне вынікаў: пераменная v (скорасць).

III. Алгарытм рашэння задачы:

1. Увод зыходных даных.

2. Згодна з формулай адлегласці: $s = vt$. Адсюль выразім v: $v = \frac{s}{t}$.

3. Вывад выніку.

IV. Апісанне пераменных:

Усе пераменныя, вызначаныя для рашэння задачы, маюць тып `real`.

Пры напісанні праграм звяртайце ўвагу на фармаціраванне іх тэксту:

- у першай пазіцыі на экране пішуць толькі словы **var**, **begin** **end**, а астатнія са зрухам на 2—4 пазіцыі ўправа;
- калі ў праграме некалькі частак, то іх можна аддзяліць адна ад адной пустым радком.

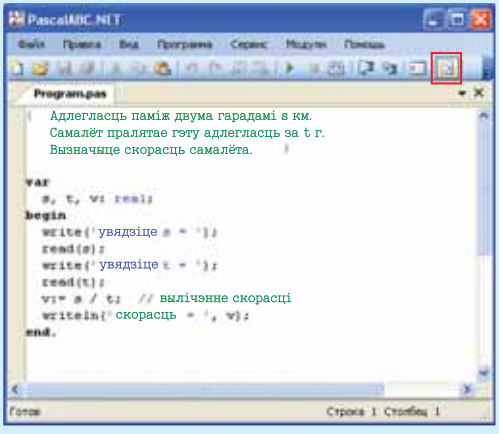
Выкананне гэтых правіл павышае чытальнасць праграмы.

У праграме можна выкарыстоўваць **каментарыі** — тэкст, які не аналізуецца пры запуску праграмы на выкананне.

Тэкст пасля знакаў // лічыцца каментарыем і вылучаецца зялёным колерам (прыклад 15.5).

У каментарыях зручна запісваць умову задачы. Тлумачэнні да каманд, запісаныя як каментарыі, патрэбныя для разумення дзеяння, выкананага камандай. У мове праграмірования Pascal каментарыі можна запісаць у некалькі радкоў. Тады тэкст, які з'яўляецца каментарыем, змяшчаюць у фігурных дужках.

Прыклад 15.5. Праграма з каментарыямі і адфармаціраваным кодам:



Значок для фармаціравання кода на панэлі інструментаў мае наступны выгляд:

1. Пералічыце этапы рашэння задачы па праграмірованню.
2. Што разумеюць пад тэсціраваннем праграмы?
3. Для чаго можна выкарыстоўваць каментарыі?

Практыкаванні

1 Дадзены x, y, z. Напішыце праграму для вылічэння значэння арыфметычных выразаў.

1. $a = \dfrac{x + y - z}{x^2 + 2}$.
2. $a = 5\dfrac{2x - z}{3 + y^2}$.
3. $a = (1 + z)\dfrac{x + \dfrac{y}{x^2 + 4}}{2 + \dfrac{1}{x^2 + 4}}$.

2 Напішыце праграму для рашэння геаметрычнай задачы.

1. Знайдзіце даўжыню акружнасці і плошчу круга зададзенага радыуса.

2*. Знайдзіце вугал пры аснове раўнабедранага трохвугольніка, калі вядомы вугал пры вяршыні.

3 Напішыце праграму для рашэння фізічнай задачы.

1. Веласіпедыст едзе з пастаяннай скорасцю v км/г. За колькі мінут ён праедзе адлегласць у s км?

2*. Аўтамабіль праходзіць першую частку шляху даўжынёй s_1 км за t_1 мін, участак шляху даўжынёй s_2 км за t_2 мін і ўчастак даўжынёй s_3 км за t_3 мін. Знайдзіце сярэднюю скорасць аўтамабіля ў км/г.

4 Напішыце праграму для рашэння хімічнай задачы.

1. У арганізме чалавека на долю атамаў кісларoду прыпадае 65 % ад масы цела. Знайдзіце масу атамаў кісларoду для сваёй масы цела.

2*. Маса атама кісларoду роўна $26.56 \cdot 10^{-27}$ (гэты лік на мове Pascal запісваецца так: 26.56E-27, літара E — англійская). Колькі атамаў кісларoду змяшчаецца ў вашым целе?

§ 16. Рэалізацыя алгарытмаў работы з цэлалікавымі данымі

У PascalABC вызначаны розныя тыпы даных для работы з цэлымі лікамі, якія дазваляюць выконваць дзеянні над данымі з розных лікавых дыяпазонаў. Чым большы дыяпазон, тым больш месца ў памяці камп'ютара адводзіцца для захоўвання пераменных.

Некаторыя цэлалікавыя тыпы даных:

Тып	Дыяпазон значэнняў
`shortint`	–128..127
`smallint`	–32768..32767
`integer`, `longint`	–2147483648..2147483647
`byte`	0..255
`word`	0..65535

16.1. Цэлалікавы тып даных

Часта пры рашэнні задач трэба працаваць з цэлымі лікамі. Для гэтага ў Pascal выкарыстоўваецца тып даных `integer`. З дапамогай пераменных гэтага тыпу задаюцца цэлыя лікі ад –2147483648 да 2147483647. Для тыпу даных `integer` вызначаны наступныя аперацыі:

Матэматычныя аперацыі	Запіс у Pascal
+ (складанне)	+
– (адніманне)	–
· (множанне)	*
цэлалікавае дзяленне	`div`
знаходжанне астачы	`mod`