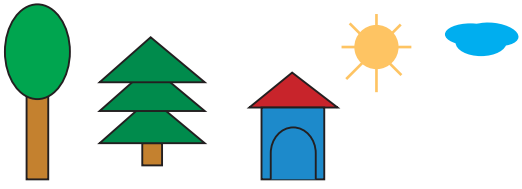
Практыкаванні

1 Падпішыце відарыс з прыкладу 14.8.

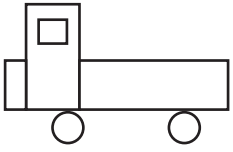
2 Дапоўніце відарыс доміка з прыкладу 14.8 відарысамі трубы і дыму з трубы ў выглядзе некалькіх авалаў:

3 Дапоўніце вынік, атрыманы пры выкананні задання 2, якімі-небудзь з прапанаваных відарысаў або прыдумайце свае.

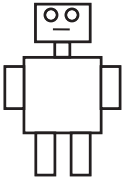


4 Напішыце праграму для стварэння відарыса. Размалюйце дадзены відарыс па сваім вырашэнні. Дадатковыя каманды для пабудовы графічных прымітываў можна знайсці ў даведачнай сістэме.

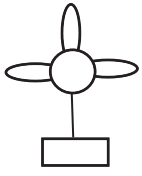
Грузавік
1



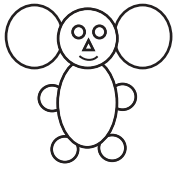
Робат
2



Кветка
3



Чабурашка
4



§ 15. Простыя і састаўныя ўмовы

15.1. Лагічны тып даных

Нагадаем вывучаныя ў 7-м класе паняцці *выказванне* і *ўмова для выканаўцы*.

Выказванне — апавядальны сказ (сцверджанне), пра які можна сказаць, праўдзівы ён ці непраўдзівы.

Умовай для выканаўцы з'яўляецца вядомае яму выказванне, якое можа выконвацца (быць праўдзівым) або не выконвацца (быць непраўдзівым).

Тып `Boolean` названы ў гонар англійскага матэматыка і логіка Джорджа Буля, які займаўся пытаннямі матэматычнай логікі ў XIX ст.

Дадзены тып прысутнічае ў пераважнай большасці моў праграміравання. У некаторых мовах рэалізуецца праз лікавы тып даных. Тады за значэнне «няпраўда» прымаецца 0, а за значэнне «праўда» — 1.

Прыклад 15.1. Прыклады лагічных выразаў:

- `3 < 7` — лагічны выраз, значэнне якога `true`;
- `2 + 2 * 2 = 8` — лагічны выраз, значэнне якога `false`;
- `abs(-5) > abs(3)` — лагічны выраз, значэнне якога `true`;
- `y >= sqr(x)` — лагічны выраз, значэнне якога можна вызначыць, толькі ведаючы значэнні пераменных x і y. Пры $x = 2$ і $y = 10$ значэнне выразу — `true`. Пры $x = 10$ і $y = 2$ — `false`.

Праверым праўдзівасць гэтых выразаў у праграме:

```
var a1, a2, a3, a4, a5: boolean;
  x, y: integer;
begin
  a1 := 3 < 7;
  writeln('a1 =', a1);
  a2 := 2 + 2 * 2 = 8;
  writeln('a2=', a2);
  a3 := abs(-5) > abs(3);
  writeln('a3 =', a3);
  x := 2; y := 10;
  a4:= y > = sqr(x);
  writeln('a4 = ', a4);
  x := 10; y := 2;
  a5 := y >= sqr(x);
  writeln('a5  =', a5);
end.
```

Вынік работы праграмы:

Окно вывода

```
a1 = True
a2 = False
a3 = True
a4 = True
a5 = False
```

Па змоўчанні `false < true`.

У мове праграміравання Pascal для работы з умовамі вызначаны **лагічны тып даных boolean.** Велічыні тыпу `boolean` могуць прымаць два значэнні — `false` (непраўдзіва) і `true` (праўдзіва).

Значэнні `false` і `true` атрымліваюцца ў выніку выканання аперацый параўнання над лікавымі данымі. Для параўнання выкарыстоўваюць знакі, паказаныя ў табліцы.

Аперацыя	PascalABC
Роўна (=)	=
Не роўна (≠)	<>
Больш (>)	>
Менш (<)	<
Больш або роўна (⩾)	>=
Менш або роўна (⩽)	<=

Параўноўваць можна канстанты, пераменныя, арыфметычныя і лагічныя выразы.

Лагічны выраз — выраз, які прымае адно з двух значэнняў: `true` або `false`.

Лагічныя выразы можна прысвойваць пераменным тыпу `boolean`, а таксама выводзіць іх значэнні на экран: будзе выведзена слова `false` або `true` адпаведна (прыклад 15.1). Умовы для выканаўцы з'яўляюцца прыватным выпадкам лагічных выразаў.

Прыклад 15.2. Напісаць праграму, якая выведзе на экран значэнне `true` або `false` у залежнасці ад таго, з'яўляецца ўведзены лік x цотным ці не.

Этапы выканання задання

I. Зыходныя даныя: x (уведзены лік).

II. Вынік: a (true або false).

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Вылічэнне значэння лагічнай пераменнай. Лік з'яўляецца цотным, калі астача ад дзялення яго на 2 роўна нулю. Значэнне пераменнай a вызначаецца значэннем выразу x **mod** 2 = 0.

3. Вывад выніку.

IV. Апісанне пераменных: x — integer, a — boolean.

Прыклад 15.2.

V. Праграма:

```
var x: integer;
  a: boolean;
begin
  write('Увядзіце x =');
  read(x);
  a := x mod 2 = 0;
  write('Лік цотны — ',a);
end.
```

VI. Тэсціраванне

Запусціць праграму і ўвесці значэнне x = 6. Вынік:

```
Окно вывода
Увядзіце x = 6
Лік цотны - True
```

Запусціць праграму і ўвесці значэнні x = 11. Вынік:

```
Окно вывода
Увядзіце x = 11
Лік цотны - False
```

15.2. Састаўныя ўмовы

З выказваннямі можна выконваць лагічныя аперацыі (**НЕ, І, АБО**). Для лагічных пераменных таксама вызначаны лагічныя аперацыі, якія адпавядаюць аперацыям над выказваннямі: **not**, **and**, **or**.

Лагічныя выразы, у якіх поруч з простымі ўмовамі (параўнаннямі) выкарыстоўваюцца лагічныя аперацыі, называюць **састаўнымі ўмовамі.**

Прывядзём табліцы праўдзівасці лагічных аперацый.

Лагічная пераменная		Вынік аперацыі		
A	*B*	**not** *A*	*A* **and** *B*	*A* **or** *B*
True	True	False	True	True
False	True	True	False	True
True	False	False	False	True
False	False	True	False	False

У мове PascalABC рэалізавана лагічная аперацыя **xor** — выключальнае **АБО**. Гэтай аперацыі адпавядае выказванне: «Толькі адно з двух выказванняў можа быць праўдзівым». Табліца праўдзівасці для аперацыі **xor**:

A	*B*	*A* **xor** *B*
True	True	False
False	True	True
True	False	True
False	False	False

Усе лагічныя аперацыі могуць выкарыстоўвацца ў дачыненні да лікаў тыпу integer. Лік разглядаецца ў двайковым уяўленні, і аперацыі выкарыстоўваюцца ў дачыненні да бітаў ліку. Біт, роўны 1, уяўляецца як праўда, а біт, роўны нулю, — як няпраўда.

Прыклад 15.3. Вызначэнне парядку дзеянняў для выразу (*a*, *d* — boolean, *c*, *b* — integer):

a **or** (c < b) **and** d

Першым выконваецца параўнанне *c* і *b*, затым лагічная аперацыя **and**, потым — **or**.

Прыклад 15.4*. Разгледзім выраз:

a < b **and** c < d

Калі *a*, *b*, *c*, *d* мае тып integer, то атрымаем памылку: «Операция '<' не применима к типам boolean и integer» (з дапамогай знака '<' нельга параўноўваць лік і лагічную пераменную). Калі пераменныя маюць тып real, то ўзнікне памылка: «Операция **'and'** не применима к типу real». Правільны запіс выразу:

(a < b) **and** (c < d)

Усе вышэйпералічаныя памылкі ўзнікаюць таму, што аперацыя and валодае большым прыярытэтам у адносінах да аперацый <. Таму спачатку будзе ажыццяўляцца спроба выканаць аперацыю b **and** c, а затым параўнання.

Прыклад 15.5. Пабудова адмаўленняў:

not not a = a;
not (a **and** b) = (**not** a) **or** (**not** b);
not (a **or** b) = (**not** a) **and** (**not** b).

Разгледзім выраз **not** a < b з пераменнымі *a* і *b* тыпу integer. Тут аперацыя not належыць да пераменнай *a*, таму ў двайковым уяўленні ліку *a* біты, роўныя 1, будуць заменены на 0, а біты, роўныя 0, — на 1. Затым атрыманы вынік параўнаецца з лікам *b*. Для адмаўлення параўнання выраз трэба запісаць так: **not** (a < b).

У лагічных выразах могуць сустракацца як арыфметычныя аперацыі, так і лагічныя. Парадак выканання аперацый вызначаецца іх прыярытэтам:

1) **not**;
2) *, /, **div**, **mod**, **and**;
3) +, –, **or**;
4) =, <>, <, >, <=, >=.

(Разгледзьце прыклад 15.3.)

Аперацыі з аднолькавым прыярытэтам выконваюцца па парадку, злева направа. Для змянення парадку выканання аперацый ужываюць дужкі (прыклад 15.4).

Пры складанні праграм часта трэба будаваць адмаўленні складаным лагічным выразам. Для гэтага карысна выкарыстоўваць тоеснасці, вядомыя з алгебры логікі (прыклад 15.5), і наступную табліцу:

Умова	Супрацьлеглая ўмова (адмаўленне ўмовы)
$a < b$	$a >= b$
$a > b$	$a <= b$
$a = b$	$a <> b$

Прыклад 15.6. Напісаць праграму, якая выдасць на экран значэнне true або false ў залежнасці ад таго, ці знаходзіцца лік *B* паміж лікамі *A* і *C*.

Этапы выканання задання

I. Зыходныя даныя: пераменныя *A*, *B*, *C* (лікі, якія ўводзяцца).

II. Вынік: rez (True або False).

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Вылічэнне значэння лагічнай пераменнай. Разгледзім два выпадкі.

Правільная няроўнасць: $A < B < C$. Гэтай няроўнасці адпавядае лагічны выраз: (A < B) **and** (B < C). Нададзім пераменнай r1 значэнне гэтага выразу.

Правільная няроўнасць: $A > B > C$. Гэтай няроўнасці адпавядае лагічны выраз: (A > B) **and** (B > C). Нададзім пераменнай r2 значэнне гэтага выразу.

Адказам на задачу будзе значэнне лагічнага выразу r1 **or** r2.

3. Вывад выніку.

IV. Апісанне пераменных: A, B, C — integer, r1, r2, rez — boolean.

Для работы з лагічнымі велічынямі могуць выкарыстоўвацца функцыі. Функцыя Ord (парадкавы нумар значэння) дазваляе пераўтвараць лагічнае значэнне ў лікавае: Ord(false) = 0, а Ord(true) = 1. Функцыі Pred (папярэдняе значэнне) і Succ (наступнае значэнне) дазваляюць пераўтвараць лагічныя значэнні:

```
D := Pred(true);   {D = false}
E := Succ(false);  {E = true}
```

Прыклад 15.6.

V. Праграма:

```
var A, B, C: integer;
 r1, r2, rez: boolean;
begin
 writeln('Увядзіце A, B, C');
 read(A, B, C);
 r1 := (A < B) and (B < C);
 r2 := (A > B) and (B > C);
 rez := r1 or r2;
 write('Лік B паміж лікамі
       A і C – ',rez);
end.
```

VI. Тэсціраванне.

Запусціць праграму і ўвесці значэнні A = 5, B = 0, C = −5. Вынік:

Окно вывода

```
Увядзіце A, B, C
5 0 -5
Лік B паміж лікамі A і C - True
```

Запусціць праграму і ўвесці значэнні A = −2, B = −7, C = 5. Вынік:

Окно вывода

```
Увядзіце A, B, C
-2 -7 5
Лік B паміж лікамі A і C - False
```

VII. Аналіз вынікаў. Для поўнага тэсціравання праграмы трэба праверыць ўсе магчымыя выпадкі ўзаемнага размяшчэння A, B, C (іх усяго 6).

1. Што такое састаўная ўмова?
2. Назавіце лагічныя аперацыі, якія выкарыстоўваюцца ў PascalABC.
3. Які прыярытэт у лагічнай аперацыі **not** (**and**, **or**)?

Практыкаванні

1 Сфармулюйце і рэалізуйце адваротную задачу для прыкладу 15.2: для ўсіх тых выпадкаў, для якіх у зыходнай задачы было true, трэба вывесці false і, наадварот, для ўсіх тых выпадкаў, у якіх у зыходнай задачы атрымлівалася false, атрымаць true.

2 У PascalABC вызначана лагічная функцыя odd(x). Значэнне гэтай функцыі true, калі лік x з'яўляецца няцотным, і false, калі x — цотны. Змяніце праграму прыкладу 15.2, выкарыстоўваючы функцыю odd.

3 Вызначыце, што робяць наступныя праграмы, і дапоўніце каманду вываду.

1.
```
var x: integer;
    a: boolean;
begin
  write('Увядзіце x =');
  read(x);
  a := x mod 10 = 0;
  write('Лік ... — ',a);
end.
```

2.
```
var x: integer;
    a: boolean;
begin
  write('Увядзіце x  =');
  read(x);
  a := (x>10) and (x < 100);
  write('Лік ... — ',a);
end.
```

4 Напішыце праграму, якая выведзе на экран значэнне `true` або `false`, у залежнасці ад таго, з'яўляецца ўведзены лік *x* дадатным ці не.

5 Напішыце праграму, якая выведзе на экран значэнне `true` або `false`, у залежнасці ад таго, з'яўляецца ўведзены лік *x* чатырохзначным ці не.

6* Зададзены два дадатныя лікі *x* і *y*. Вызначыце, ці дакладна, што першы лік меншы за другі і хоць бы адзін з іх няцотны. Выведзіце на экран `true` або `false`.

§ 16. Аператар галінавання

Выкарыстанне кіруючых канструкцый прадугледжвае запіс праграмы ў структураваным выглядзе. Структураванасць праграм дасягаецца за кошт водступаў, якія рэгулююць змяшчэнне ўкладзеных алгарытмічных канструкцый.

Можна выконваць наступнае правіла: пры руху курсора ўніз ад «пачатку» структуры да яе «канца» на шляху курсора могуць сустрэцца толькі прабелы. Усё, што знаходзіцца «ўнутры» структуры, змяшчаецца правей.

Кнопка дазваляе пераўтварыць код праграмы да структураванага выгляду.

Прыклад 16.1.
V. Праграма:
```
var x: integer;
begin
 write('Увядзіце x = '); read(x);
 if x > 0 then
   write('дадатнае')
 else
   write('не дадатнае');
end.
```

16.1. Запіс аператара галінавання

Алгарытмічная канструкцыя *галінаванне* (гл. блок-схему ў прыкладзе 13.2, с. 60) забяспечвае выкананне той ці іншай паслядоўнасці каманд у залежнасці ад праўдзівасці або непраўдзівасці некаторай умовы.

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

Для запісу аператара галінавання выкарыстоўваюць каманды **if**. Фармат каманды:
```
if <умова> then
begin
  Каманды 1;
end
else
begin
  Каманды 2;
end;
```

Аператар галінавання можа быць у поўнай або ў скарочанай формах. У скарочанай форме адсутнічае блок else:

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

Умова ў запісе аператара галінавання бывае простай і састаўной. Аператарныя дужкі могуць быць апушчаны, калі ўнутры іх знаходзіцца адна каманда.

Прыклад 16.1. Зададзены лік x. Трэба вызначыць, з'яўляецца ён дадатным ці не, і вывесці адпаведнае паведамленне.

Этапы выканання задання

I. Зыходныя даныя: x (уведзены лік).

II. Вынік: адпаведнае паведамленне.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Праверка значэння выразу ($x > 0$).

3. Вывад выніку.

IV. Апісанне пераменных: x — integer.

16.2. Рашэнне задач з выкарыстаннем аператара галінавання

Прыклад 16.2. У момант часу 00:00 на святлафоры для пешаходаў уключылі зялёны сігнал. Далей сігнал святлафора змяняецца кожную мінуту: 1 мінуту гарыць зялёны сігнал, 1 мінуту — чырвоны. Вядома, што з моманту ўключэння святлафора прайшло m мінут. Патрабуецца нарысаваць святлафор з уключаным сігналам у

Прыклад 16.1. *Працяг.*

VI. Тэсціраванне.

Запусціць праграму і ўвесці значэнне $x = 5$. Вынік:

Окно вывода

```
Увядзіце x = 5
дадатны
```

Запусціць праграму і ўвесці значэнне $x = -1$. Вынік:

Окно вывода

```
Увядзіце час  x = -1
не дадатны
```

VII. Аналіз вынікаў. Для поўнай праверкі праграмы патрабуецца яшчэ праверыць значэнне $x = 0$.

Окно вывода

```
Увядзіце x = 0
не дадатны
```

Прыклад 16.2.

V. Праграма:

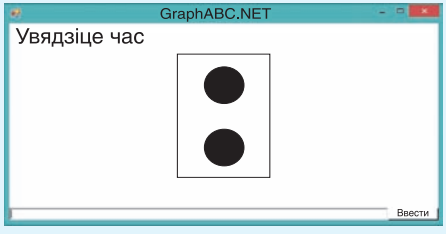
```
uses GraphABC;
var m:integer;
begin
  Rectangle(250,50,390,250);
  SetBrushColor(clBlack);
  Circle(320,100,30);
  Circle(320,200,30);
  SetBrushColor(clWhite);
  writeln('Увядзіце час');
  read(m);
  writeln(m)[1];
  if m mod 2 = 1 then
    FloodFill(320,100,clRed)
  else
    FloodFill(320,200,clGreen);
end.
```

[1] Пры ўводзе даных у графічным акне яны не адлюстроўваюцца на экране. Для таго каб бачыць, што ўвялі, неабходна дадаткова вывесці ўведзенае значэнне.

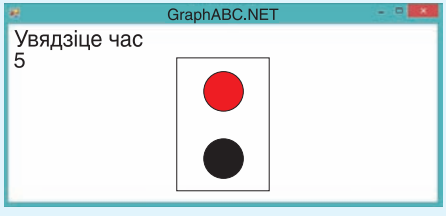
Прыклад 16.2. *Працяг.*

VI. Тэсціраванне.

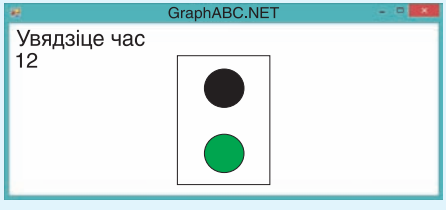
Выгляд графічнага акна да ўводу ліку:



Увесці значэнне $x = 5$. Вынік:



Увесці значэнне $x = 12$. Вынік:



Прыклад 16.3.

V. Праграма:

```
Var x1, y1, x2, y2, r_T, r_K: real;
begin
  writeln('Танін дом');
  read(x1,y1);
  writeln('Кацін дом');
  read(x2,y2);
  r_T:= sqrt(x1*x1+y1*y1);
  r_K:= sqrt(x2*x2+y2*y2);
  if r_T < r_K then
    writeln('Танін дом бліжэй')
  else
    writeln('Кацін дом бліжэй');
end.
```

VI. Запусціць праграму і ўвесці значэнні: Танін дом — $x1 = 2.3$, $y1 = 4.5$, Кацін дом — $x2 = -2.1$, $y2 = 4.9$ адпаведнасці з уведзеным значэннем часу.

Этапы выканання задання

I. Зыходныя даныя: m (зададзены час).

II. Вынік: рысунак святлафора, які залежыць ад значэння m.

III. Алгарытм рашэння задачы.

1. Рысаванне святлафора (прамавугольнік і 2 кругі) з выключанымі сігналамі.

2. Увод зыходных даных.

3. Колер сігналу будзе залежаць ад таго, цотным ці няцотным будзе значэнне m. Калі m цотнае — сігнал зялёны (зафарбоўваем ніжні круг), калі няцотнае — чырвоны (зафарбоўваем верхні круг).

4. Зафарбуем патрэбны круг колерам у залежнасці ад цотнасці m.

IV. Апісанне пераменных: m — `integer`.

Прыклад 16.3. Таня і Каця жывуць у розных дамах. Ім стала цікава, хто з іх жыве бліжэй да школы. Яны змясцілі на карце прамавугольную сістэму каардынат так, каб школа мела каардынаты (0; 0). Вядома, што Танін дом мае каардынаты (x1; y1), а Кацін (x2; y2). Дзяўчынкі ходзяць у школу па прамой і праходзяць розныя адлегласці. Трэба напісаць праграму, якая вызначыць, чый дом бліжэй да школы.

Этапы выканання задання

I. Зыходныя даныя: каардынаты дамоў дзяўчынак $x1$, $y1$, $x2$, $y2$.

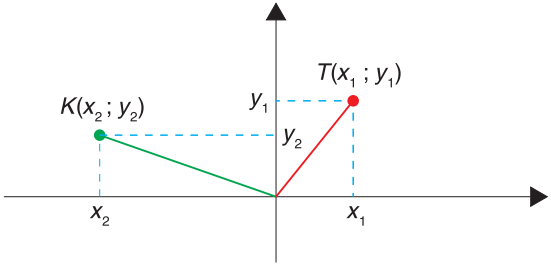
II. Вынік: паведамленне пра тое, чый дом бліжэй.

III. Алгарытм рашэння задачы.

1. Увод каардынат дамоў.

2. Вылічэнне адлегласцей да школы: *r_T* (ад Танінага дома) і *r_K* ад Кацінага дома). Для вылічэння выкарыстаем тэарэму Піфагора:

$$r\_T = \sqrt{x_1^2 + y_1^2} \text{ і } r\_K = \sqrt{x_2^2 + y_2^2}.$$



3. Параўнанне адлегласцей. Вывад адказу.

IV. Апісанне пераменных: *x*1, *y*1, *x*2, *y*2, *r_T*, *r_K* маюць тып `real`.

Прыклад 16.4. Вася пачаў займацца стральбой з лука. Для трэніроўкі ён вырашыў стварыць мадэль мішэні, якая будзе рэагаваць на лазер. Мішэнь уяўляе сабой два кругі (страляе Вася пакуль не вельмі добра) рознага радыуса з агульным цэнтрам. Калі Вася пападае у маленькі круг, то круг загараецца зялёным. Вялікі круг пры пападанні ў яго загараецца жоўтым. Калі Вася не папаў ні ў адзін з кругоў, то вобласць па-за кругамі загараецца чырвоным. Неабходна рэалізаваць камп'ютарную мадэль Васевай мішэні (пры пападанні на мяжу круга нічога не павінна адбывацца).

Этапы выканання задання

I. Выходныя даныя: каардынаты пункта стрэлу (*x*; *y*).

II. Вынік: рысунак мішэні.

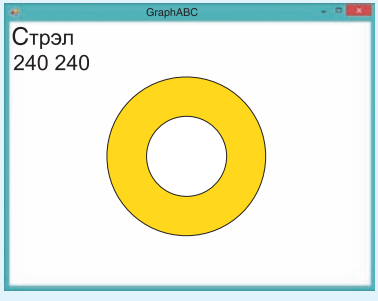
III. Алгарытм рашэння задачы.

Прыклад 16.3. *Працяг.*
Вынік павінен быць такім:

Окно вывода
```
Танін дом
2.3 4.5
Кацін дом
-2.1 4.9
Танін дом бліжэй
```

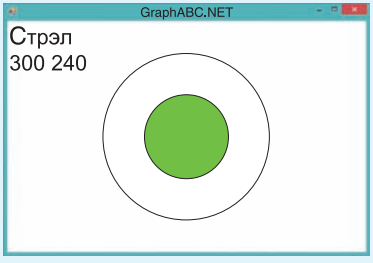
Прыклад 16.4.
V. Праграма:
```
uses GraphABC;
var x,y, x0, y0, R_b, R_m, z:
integer;
begin
  x0 := 320; y0 := 240;
  R_b := 150; R_m := 75;
  Circle(x0,y0,R_b);
  Circle(x0,y0,R_m);
  writeln('Стрэл');
  read(x,y);
  writeln(x,' ',y);
  z := sqr(x-x0)+sqr(y-y0);
  if z < sqr(R_m) then
    FloodFill(x,y,clLightGreen)
  else
    if z < sqr(R_b) then
      FloodFill(x,y,clYellow)
    else
      FloodFill(x,y,clRed);
end.
```
VI. Тэсціраванне.

Запусціць праграму і ўвесці каардынаты стрэлу (240; 240). Вынік:

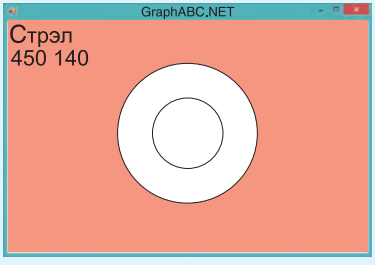


Прыклад 16.4. *Працяг.*

Запусціць праграму яшчэ раз і ўвесці каардынаты стрэлу (300; 240).



Запусціць праграму яшчэ раз і ўвесці каардынаты стрэлу (450; 140).



Прыклад 16.5.

V. Праграма:

```
var a, a1, a2, a3: integer;
begin
  write('Увядзіце a = ');
  read(a);
  if (a > 99) and (a < 1000) then
  begin
    //Першая лічба
    a1 := a div 100;
    //Другая лічба
    a2 := a mod 100 div 10;
    //Трэцяя лічба
    a3 := a mod 10;
    writeln(a1);
    writeln(a2);
    writeln(a3);
  end
  else
    writeln('не трохзначны');
end.
```

1. Рысаванне мішэні: 2 кругі радыусаў $R\_b = 150$ і $R\_m = 75$ з цэнтрам у пункце $(x_0; y_0)$, $x_0 = 320$, $y_0 = 240$. Спачатку рысуем круг большага радыуса.

2. Увод даных: каардынаты пункта стрэлу.

3. Колер рысунка будзе залежаць ад таго, у якую вобласць адносна кругоў трапіў пункт. Магчымы 3 выпадкі:

1) пункт унутры маленькага круга. Даўжыня адрэзка паміж пунктам і цэнтрам круга меншы за радыус. Па тэарэме Піфагора:

$(x - x_0)^2 + (y - y_0)^2 < R\_m^2$;

2) калі ўмова а) не выконваецца, правяраем, ці належыць пункт вялікаму кругу:

$(x - x_0)^2 + (y - y_0)^2 < R\_b^2$;

3) калі ўмовы а) і б) не выконваюцца, то Вася не трапіў у мішэнь.

4. Для скарачэння запісу вызначым пераменную $z = (x - x_0)^2 + (y - y_0)^2$.

5. Закрасім патрэбную вобласць колерам у залежнасці ад праверкі ўмоў.

IV. Апісанне пераменных: x, y, x_0, y_0, R_b, R_m, z маюць тып `integer`.

Прыклад 16.5. Праверыць, ці з'яўляецца ўведзены лік трохзначным, і калі так, то вывесці лічбы гэтага ліку ў асобных радках.

Этапы выканання задання

I. Зыходныя даныя: a (трохзначны лік).

II. Вынік: пераменныя $a1$, $a2$, $a3$ (лічбы ліку) або паведамленне «не трохзначны».

III. Алгарытм рашэння задачы.

1. Увод зыходнага ліку.

2. Праверка ліку. Лік a з'яўляецца трохзначным, калі $99 < a < 1000$.

3. Калі лік трохзначны, вылучаем яго лічбы:

1) для вылучэння першай лічбы $a1$ знаходзім цэлую частку ад дзялення ліку a на 100;

2) для вылучэння другой лічбы $a2$ ліку a знаходзім астачу ад яго дзялення на 100, а затым цэлую частку ад дзялення атрыманай астачы на 10;

3) апошняя лічба ліку $a3$ з'яўляецца астачай ад дзялення ліку a на 10.

4. Вывад выніку.

IV. Апісанне пераменных: усе пераменныя маюць тып `integer`.

Прыклад 16.5. *Працяг.*

VI. Тэсціраванне.

Запусціць праграму і ўвесці значэнне 345.

Вынік наступны:

```
Окно вывода
Увядзіце a = 345
3
4
5
```

Іншыя варыянты зыходных даных:

```
Окно вывода
Увядзіце a = 24
не трохзначны
```

1. Што такое аператар галінавання?
2. Чым адрозніваецца поўны запіс аператара галінавання ад скарочанага?
3. Ці можна выкарыстоўваць састаўныя ўмовы ў аператары галінавання?

Практыкаванні

1 Ці можна змяніць лагічны выраз у аператары галінавання ў прыкладзе 16.1 так, каб паведамленні `'дадатны'` і `'не дадатны'` прыйшлося памяняць месцамі? Калі так, то як гэта зрабіць?

2* Якія змяненні трэба ўнесці ў праграму прыкладу 16.1, каб для ліку разглядаліся тры выпадкі: `'дадатны'`, `'адмоўны'`, `'роўны нулю'`?

3 Падключыце графічны рэжым у праграме прыкладу 16.1. Змяніце праграму так, каб паведамленне `'дадатны'` выводзілася чырвоным колерам, а паведамленне `'не дадатны'` — сінім.

4* Змяніце праграму ў прыкладзе 16.2 так, каб цотнасць (няцотнасць) ліку правяралася з выкарыстаннем функцыі odd.

5 Напішыце праграму. Зададзены лік x. Калі лік цотны, то нарысаваць на экране зялёны прамавугольнік, а калі няцотны, то чырвоны круг (гл. прыклад 16.2).

6 Дабаўце ў праграму з прыкладу 16.3 праверку карэктнасці зыходных даных: каардынаты дамоў павінны быць такімі, каб адлегласці да школы былі рознымі. Калі адлегласці аднолькавыя, то вывесці паведамленне `'Каардынаты ўведзены няправільна'`, а калі розныя, то вывесці адказ.

7 Якія змяненні спатрэбіцца ўнесці ў праграму з прыкладу 16.3, калі дапусціць, што дзяўчынкі могуць праходзіць аднолькавыя адлегласці? Унясіце змяненні ў праграму і праверце правільнасць яе работы.

8 Для ўскладнення трэніровак Вася (прыклад 16.4) вырашыў змяняць месцазнаходжанне мішэні і радыусы кругоў. Дабаўце ў праграму магчымасць уводу радыусаў вялікага і маленькага кругоў, а таксама цэнтра мішэні. Праверце правільнасць работы праграмы на розных наборах зыходных даных.

9 Як вядома, шмат якія задачы маюць не адзінае рашэнне. Так, Юля знайшла іншы спосаб вылічэння другой лічбы трохзначнага ліку для прыкладу 16.5. Якую з каманд выкарыстоўвала Юля? Растлумачце, што атрымаецца пры выкананні кожнай з прыведзеных каманд.

1) `a2:=a mod 10 div 10;` 2) `a2:=a div 10 mod 10;` 3) `a2:=a div 100 mod 10.`

10 Праграму з прыкладу 16.5 змянілі. Сфармулюйце ўмову задачы, якая рашаецца з дапамогай гэтай праграмы.

```
var a, a1, a2, a3: integer;
begin
 write('Увядзіце  a = '); read(a);
 if (a > 99) and (a < 1000) then
 begin
  // Першая  лічба
  a1 := a div 100;
  // Другая  лічба
  a2 := a mod 100 div 10;
  // Трэцяя  лічба
  a3 := a mod 10;
  if a1 mod 2 = 0 then
     writeln(a1,'– цотная ');
  if a2 mod 2 = 0 then
     writeln(a2,'– цотная ');
  if a3 mod 2 = 0 then
     writeln(a3, '– цотная);
  if odd(a1) and odd(a2) and odd(a3) then
     writeln('няма  цотных  лічбаў');
 end
 else
  writeln('не  трохзначны');
end.
```

11 Праграму з задання 10 праверылі для некаторых выпадкаў. Ці ўсе магчымыя сітуацыі разгледзелі? Што трэба дабавіць?

Окно вывода	Окно вывода	Окно вывода	Окно вывода
Увядзіце a = 246 2 - цотная 4 - цотная 6 - цотная	Увядзіце a = 103 0 - цотная	Увядзіце a = 537 няма цотных лічбаў	Увядзіце a = 26 не трохзначны

12 Пеця вырашыў удасканаліць праграму з задання 10 і праверку лічбаў у ліку запісаў наступным чынам:

```
if a1 mod 2 = 0 then
 writeln(a1,' – цотная')
else
 if a2 mod 2 = 0 then
   writeln(a2,' – цотная')
 else
   if a3 mod 2 = 0 then
     writeln(a3, ' – цотная')
   else
     writeln('няма цотных лічбаў');
```

Чаму Пецева адзнака аказалася невысокай? Прывядзіце прыклады, для якіх праграма выдае няправільны адказ. Прывядзіце прыклады, у якіх праграма выдае правільны адказ, калі такое магчыма.

13 Дадзены натуральны лік. Напішыце праграму, якая правярае, ці з'яўляецца ён трохзначным і ці кратная 7 сума яго лічбаў.

14* Дадзены натуральны лік. Напішыце праграму, якая правярае, ці з'яўляецца ён чатырохзначным і ці размешчаны яго лічбы ў парадку змяншэння.

15* Вася навучыўся трапляць у цэнтр мішэні з прыкладу 16.4 і вырашыў перайсці да больш складаных трэніровак. Цяпер яго мішэнь уяўляе сабой тры ўкладзеныя кругі з радыусамі $R1$, $R2$, $R3$ (вядома, што $R1 < R2 < R3$). Рэалізуйце камп'ютарную мадэль гэтай мішэні. Колеры выбярыце самастойна.

§ 17. Аператар цыкла

17.1. Аператар цыкла з перадумовай

Алгарытмічная канструкцыя *паўтарэнне* (*цыкл*) уяўляе сабой паслядоўнасць дзеянняў, якія выконваюцца шматразова (гл. блок-схему ў прыкладзе 13.2, с. 60). Саму паслядоўнасць называюць **целам цыкла.**

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

У Pascal існуюць розныя магчымасці кіраваць тым, колькі разоў будзе паўтарацца цела цыкла. Можа быць зададзена ўмова працягвання або

Цыкл з зададзенай умовай заканчэння работы ў PascalABC запісваецца наступным чынам:

```
repeat
  цела цыкла;
until <умова>;
```

Цыкл працуе, пакуль умова непраўдзівая, і спыняе работу, калі ўмова становіцца праўдзівай.

Гэты цыкл называюць **цыклам з постумовай**, паколькі праверка ўмовы ажыццяўляецца пасля выканання цела цыкла. Цыкл з постумовай заўсёды выконваецца хоць бы адзін раз.

Цыклы `repeat` і `while` ў PascalABC узаемазамяняльныя, таму пры напісанні праграм дастаткова выкарыстання толькі аднаго з іх.

Прыклад 17.1.

V. Праграма:

```
uses GraphABC;
var x, y, r: integer;
begin
  r := 10;
  x := 10; y := 10;
  while x < 640 do
  begin
    Circle(x, y, r);
    x := x + 20;
  end;
end.
```

VI. Тэсціраванне.

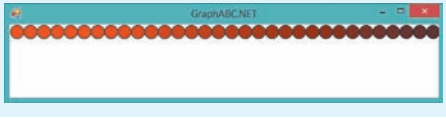
Запусціць праграму. Вынік:

VII. Лікавае значэнне ва ўмове цыкла можна замяніць функцыяй, якая вызначае гарызантальнае разрашэнне акна: `WindowWidth`:

```
while x < WindowWidth do
```

Функцыі `RedColor`, `GreenColor`, `BlueColor` дазваляюць мяняць інтэнсіўнасць адпаведнага колеру.

```
uses GraphABC;
var x, y, r, c: integer;
begin
  r := 10;
  x := 10; y := 10;
  c := 255;
  while x < 640 do
  begin
   //Інтэнсіўнасць чырвонага
    SetBrushColor(RedColor(c));
    Circle(x,y,r);
    x := x + 20;
   //Памяншэнне інтэнсіўнасці
    c := c-5;
  end;
end.
```



заканчэння работы цыкла, а таксама лік паўтарэнняў цела цыкла.

Цыкл з перадумовай выкарыстоўваецца ў тым выпадку, калі вядома ўмова працягвання работы. Для запісу аператара цыкла з перадумовай выкарыстоўваецца каманда `while`. Фармат каманды:

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

Прыклад 17.1. Напісаць праграму для рысавання рада акружнасцей з радыусам 10 пікселей уздоўж верхняга краю графічнага акна.

Этапы выканання задання

I—II. Вынікам работы праграмы, якая не залежыць ад зыходных даных, будзе рысунак, што адлюстроўвае рад акружнасцей уздоўж верхняга краю графічнага акна.

III. Алгарытм рашэння задачы.

1. *Становішча першай акружнасці*. Акружнасць размесцім ў верхнім левым вугле. Для гэтага задаецца радыус $r = 10$ і каардынаты цэнтра $x = 10$, $y = 10$.

2. *Становішча любой іншай акружнасці*, якая задавальняе ўмову задачы, будзе залежаць ад кардынаты x. У цыкле будзем змяняць значэнне x. Кожнае новае значэнне будзе на 20 (на памер дыяметра) большым за папярэдняе.

3. Цыкл павінен завяршыцца, калі значэнне каардынаты x стане большым, чым 640 — гарызантальны памер акна.

IV. Апісанне пераменных: x, y, r — `integer`.

17.2. Аператар цыкла з параметрам

Цыкл з параметрам выкарыстоўваецца тады, калі вядома колькасць паўтарэнняў.

Для запісу аператара цыкла з параметрам выкарыстоўваецца каманда **for**. Фармат каманды:

```
for var[1] i:= N1 to N2 do
begin
  цела цыкла;
end;
```

Або

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

У першым варыянце параметр цыкла i змяняецца ад *N1* да *N2*, кожны раз павялічваючыся на 1. У другім — параметр i памяншаецца на 1 пры кожным выкананні цела цыкла ад *N2* да *N1*. Калі *N1* > *N2*, цыкл не выконваецца ні разу. Змяняць значэнне параметра ўнутры цела цыкла нельга.

Прыклад 17.2. Напісаць праграму для вываду табліцы множання на зададзены лік *x*.

Этапы выканання задання

I. Зыходныя даныя: *x* (уведзены лік).

II. Вынік: 9 радкоў выгляду *a* * *x* = *c*.

III. Алгарытм рашэння задачы.

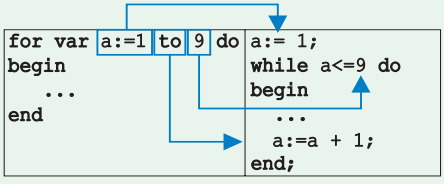
1. Значэнне пераменнай *a* змяняецца ў цыкле ад 1 да 9.

2. Значэнне пераменнай $c = a \cdot x$.

3. Паколькі колькасць паўтарэнняў загадзя вядомая, выкарыстаем цыкл for.

IV. Апісанне пераменных: *x*, *c* — integer.

Любы цыкл **for** можа быць заменены на цыкл **while**:



Адваротнае не заўсёды магчыма.

Прыклад 17.2.

V. Праграма:

```
var x, c : integer;
begin
 write('Увядзіце x = '); read(x);
 for var a : = 1 to 9 do
 begin
  c := a * x;
  writeln(a,' * ',x,' = ',c);
 end;
end.
```

VI. Тэсціраванне.

Запусціць праграму. Увесці $x = 7$.

Окно вывода

```
Увядзіце x = 7
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
```

Рашэнне з дапамогай цыкла while:

```
var a, x, c : integer;
begin
 write('Увядзіце x = '); read(x);
 a := 1;
 while a < = 9 do
 begin
  c := a * x;
  writeln(a,' * ',x,'= ',c);
  a := a + 1;
 end;
end.
```

VII. Праверыць вынік.

[1] Ключавое слова var можа быць прапушчана, тады пераменная і павінна быць апісана (як integer) у раздзеле апісання var да пачатку праграмы.

Пры рашэнні задач з выкарыстаннем аператара цыкла важна правільна выбраць від цыкла. Калі вядома колькасць паўтарэнняў цела цыкла, то выбіраюць цыкл **for**, а інакш — цыкл **while**.

Унутры цыкла можна выкарыстоўваць аператары break (неадкладны выхад з бягучага цыкла) і аператар continue (пераход да канца цела цыкла).

Прыклад 17.3.

V. Праграма:

```
uses GraphABC;
var a,x1,y1,x2,y2: integer;
begin
  write('Увядзі  a = ');
  read(a); write(a);
  x1 := 50; y1 := 50;
  x2 := 450; y2 := 450;
  for var i := 1 to 20  do
  begin
    Rectangle(x1,y1, x2,y2);
    x1 := x1 + a;   y1 := y1 + a;
    x2 := x2 - a;   y2 := y2 - a;
  end;
end.
```

VI. Тэсціраванне.

Запусціць праграму і ўвесці значэнне $a = 10$. Вынік:

17.3. Рашэнне задач з выкарыстаннем аператара цыкла

Прыклад 17.3. Нарысаваць 20 квадратаў з агульным цэнтрам. Даўжыня стараны самага вялікага квадрата 400, верхні левы вугал размешчаны ў пункце (50; 50). Каардынаты верхняга левага і ніжняга правага вуглоў кожнага наступнага квадрата змяняюцца на a (a — уводзіцца).

Этапы выканання задання

I. Зыходныя даныя: a (уведзены лік).

II. Вынік: рысунак, які адлюстроўвае квадраты.

III. Алгарытм рашэння задачы.

1. Першым рысуецца самы вялікі квадрат. Каардынаты яго верхняга левага вугла $x1 = 50$, $y1 = 50$. Каардынаты ніжняга правага вугла $x2 = 450$, $y2 = 450$.

2. Для вызначэння становішча іншага квадрата трэба каардынаты верхняга левага вугла павялічыць на a, а ніжняга правага — паменшыць на a.

3. Будзем выкарыстоўваць цыкл for, паколькі зададзена колькасць квадратаў.

IV. Апісанне пераменных: a, $x1$, $y1$, $x2$, $y2$ — integer.

Прыклад 17.4*. Вывесці на экран найбольшы натуральны лік з прамежка $[n, m]$, які дзеліцца на зададзены лік x.

Этапы выканання задання

I. Зыходныя даныя: n, m (межы прамежка), x (зададзены лік).

II. Вынік: шуканы лік або паведамленне «Няма такіх лікаў».

III. Алгарытм рашэння задачы.

1. Няхай i — гэта бягучы лік з прамежка.

2. Паколькі нас цікавіць найбольшы лік з прамежка, то прагляд лікаў пачнём са значэння $i = m$. На кожным кроку будзем памяншаць i на 1.

3. Цыкл завяршыцца, калі мы знайшлі лік, які дзеліцца на x без астачы (астача роўна нулю), або прагледзелі ўсе лікі з прамежка $[n, m]$.

4. Паколькі колькасць паўтарэнняў загадзя невядомая, выкарыстаем цыкл `while`.

Цыкл будзе працягваць работу да таго часу, пакуль умова, сфармуляваная ў пункце 3, будзе непраўдзівай. А менавіта: непраўдзівай павінна быць умова `(i < n) or (i mod x = 0)`. Тады ўмова `not ((i < n) or (i mod x = 0))` будзе праўдзівай. Згодна з правіламі пабудовы адмаўленняў (гл. прыклад 15.5) гэту ўмову можна замяніць умовай: `(i >= n) and (i mod x <> 0)`. Яе і будзем выкарыстоўваць у якасці ўмовы цыкла.

5. Калі па заканчэнні цыкла $i = n - 1$, то няма лікаў, якія задавальняюць умову задачы.

IV. Апісанне пераменных: n, m, x, i — `integer`.

Прыклад 17.4*.

V. Праграма:

```
var i, n, m, x : integer;
begin
 writeln('Увядзіце межы n, m');
 read(n,m);
 write('Увядзіце  x = ');
 read(x);
 i := m;
 while (i >= n) and
       (i mod x <> 0) do
  i := i - 1;
 if i = n - 1 then
  writeln('Няма такіх лікаў')
 else
  writeln('Шуканы лік – ',i);
end.
```

VI. Тэсціраванне.

Запусціць праграму і ўвесці значэнні $n = 10$, $m = 20$, $x = 3$. Вынік:

Окно вывода

```
Увядзіце межы n, m
10 20
Увядзіце x = 3
Шуканы лік - 18
```

Запусціць праграму і ўвесці значэнні $n = 38$, $m = 45$, $x = 37$. Вынік:

Окно вывода

```
Увядзіце межы n, m
38 45
Увядзіце x = 37
Няма такіх лікаў
```

1. Што такое аператар цыкла?
2. Якім чынам можна кіраваць колькасцю выкананняў цела цыкла?
3. Як запісваецца аператар цыкла з перадумовай?
4. Як запісваецца аператар цыкла з параметрам?

Практыкаванні

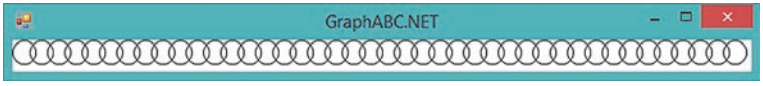
1 Змяніце праграму з прыкладу 17.1.

1. Радыусы акружнасцей роўны 20.
2. Акружнасці размяшчаюцца ўздоўж левага краю акна.
3. Радыус акружнасці ўводзіцца карыстальнікам.

4. Акружнасці ўтвараюць рамку вакол акна.

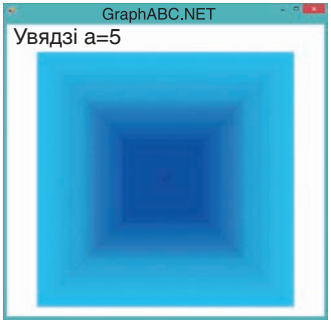
5*. Карыстальнік задае мяжу акна, уздоўж якой будуць размяшчацца акружнасці (напрыклад: 1 — верхняя, 2 — левая, 3 — правая, 4 — ніжняя).

2 Якія змяненні трэба ўнесці ў праграму з прыкладу 17.1 для таго, каб рысунак выглядаў наступным чынам?



3 Унясіце змяненні ў праграму з прыкладу 17.2. Карыстальнік задае значэнне другога множніка, а таксама пачатковае і канчатковае значэнні першага множніка.

4 Пры якім максімальным значэнні *a* на экране будуць бачныя ўсё 20 квадратаў з прыкладу 17.3? Чаму пры вялікіх значэннях *a* не бачныя ўсе квадраты? Змяніце праграму так, каб квадраты адлюстроўваліся ад самага маленькага да самага вялікага (устанавіце празрыстую заліўку).



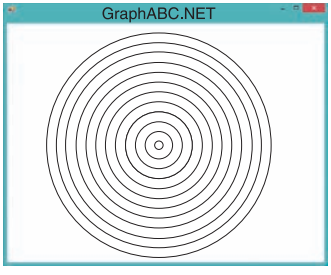
5 Якія змяненні трэба ўнесці ў праграму з прыкладу 17.3, каб атрымаць наступны відарыс? Функцыі для змянення інтэнсіўнасці колеру гл. у прыкладзе 17.1.

6 Змяніце праграму з прыкладу 17.3. Даўжыня стараны самага вялікага квадрата 400, а даўжыня стараны кожнага наступнага квадрата на *x* меншая (*x* уводзіцца).

7 Напішыце праграму, якая рысуе шэраг акружнасцей зададзенага радыуса, размешчаных па дыяганалі графічнага акна. Разгледзьце два варыянты:

1. Графічнае акно квадратнае.

2*. Графічнае акно прамавугольнае.



8 Напішыце праграму, якая рысуе канцэнтрычныя акружнасці з цэнтрам у сярэдзіне графічнага акна. Радыус самай маленькай акружнасці — 10 пікселяў. Розніца радыусаў — 20 пікселяў. Выкарыстоўвайце змяненне інтэнсіўнасці якога-небудзь колеру (або двух адначасова) для заліўкі кругоў.

9 У магазіне прадаюць цукеркі ва ўпакоўках па 0.1 кг, 0.2 кг, ... 0.9 кг, 1 кг. Вядома, што 1 кг цукерак каштуе *x* рублёў. Выведзіце кошты кожнай упакоўкі ў выглядзе:

0.1 кг цукерак каштуе ... р.;

0.2 кг цукерак каштуе ... р. ... .

10* Выведзіце на экран такі найменшы натуральны лік з прамежка [*n*, *m*], які з'яўляецца няцотным і не дзеліцца на ўведзенае значэнне *x*.