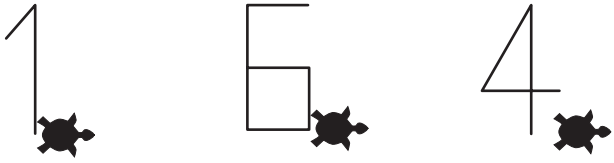
6 Напішыце каманды для пабудовы відарысаў:

а) лічба 1; б) літара Б; в) лічба 4.



Пасля малявання перамясціце *Чарапаху* на 10 направа ад ніжняга правага вугла і павярніце яе так, каб яна глядзела направа.

§ 19. Вывучэнне і змяненне гатовых праграм

Прыклад 19.1. Заданне колеру. Па змоўчанні задаецца рэжым устаноўкі колеру з выкарыстаннем тэкставых значэнняў. У 15-м радку праграмы рэжым змяняецца. Цяпер колер можна задаваць лікавымі значэннямі. У 24-м радку вяртаецца спосаб задання колеру па змоўчанні. У 25-м радку ў адной камандзе задаецца колер лініі і колер заліўкі.

Праграма

```
1. import turtle
2. turtle.shape('turtle')
3. turtle.pensize(2)
4. turtle.penup()
5. turtle.setpos(-100,0)
6. turtle.pendown()
7. turtle.color('darkred')
8. turtle.fillcolor('tomato')
9. turtle.begin_fill()
10. turtle.circle(50)
11. turtle.end_fill()
```

19.1. Дадатковыя каманды *Чарапахі*

Пры маляванні *Чарапаха* можа змяняць колер ліній і заліваць намаляваныя фігуры. Для гэтага выкарыстоўваюцца каманды `color(x)` і `fillcolor(x)` адпаведна. Замест зменнай `x` трэба задаць значэнне колеру. Яго можна задаваць з дапамогай англійскіх назваў колераў, якія запісваюцца ў двукоссі. Гэты спосаб задаецца па змоўчанні. Калі трэба задаць колер лічбавымі значэннямі, то можна змяніць рэжым уводу колеру з дапамогай каманды `colormode(255)`. Паглядзець гэтыя значэнні можна ў графічным рэдактары Paint. Для звароту рэжыму ўводу колеру назвамі выкарыстоўваецца каманда `colormode (1.0)`.

Для задання колеру ліній і колеру заліўкі можна выкарыстоўваць дзве розныя каманды або адну з двума параметрамі: `color(x,y)`. Першы параметр `x` вызначае колер лініі, а другі `y` — колер заліўкі. У прыкладзе 19.1 паказаны розныя спосабы задання колеру і змянення рэжыму адлюстравання колераў. Паглядзець тэкставыя і лікавыя значэнні колеравых канстант можна ў дадатковых матэрыялах.

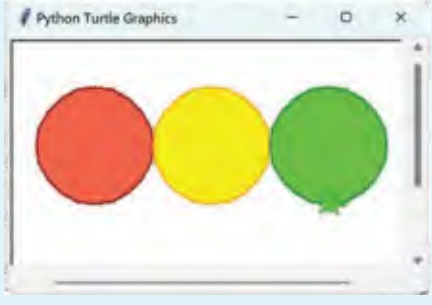
Акрамя каманд, якія былі разгледжаны ў папярэднім параграфе, у выканаўцы *Чарапаха* ёсць і іншыя. У табліцы прыведзены магчымыя каманды выканаўцы *Чарапаха*.

Каманда	Дзеянне
`pensize(x)`	Змяніць таўшчыню лініі, якую малюе *Чарапаха*
`begin_fill()`	Каманда прапісваецца перад камандамі малявання фігуры
`end_fill()`	Заліць фігуру, якая намалявана з дапамогай каманд, размешчаных паміж `begin_fill()` і `end_fill()`

Прыклад 19.1. *Працяг.*

```
12. turtle.penup()
13. turtle.setpos(0,0)
14. turtle.pendown()
15. turtle.colormode(255)
16. turtle.color(255,165,0)
17. turtle.fillcolor(255,255,0)
18. turtle.begin_fill()
19. turtle.circle(50)
20. turtle.end_fill()
21. turtle.penup()
22. turtle.setpos(100,0)
23. turtle.pendown()
24. turtle.colormode(1.0)
25. turtle.color('green','lime')
26. turtle.begin_fill()
27. turtle.circle(50)
28. turtle.end_fill()
```

Вынік работы праграмы



У табліцы пералічаны не ўсе каманды выканаўцы *Чарапаха*. Яшчэ яна можа выконваць каманды: `dot()`, `stamp()`, `speed()`, `clear()` і інш. Аб тым, якія дзеянні выконваюць гэтыя каманды, можна даведацца ў даведачнай сістэме.

Прыклад 19.2. Напрамак руху *Чарапахі*.

Вугал павароту	Напрамак *Чарапахі*
0	Направа, на ўсход
90	Уверх, на поўнач
180	Налева, на захад
270	Уніз, на поўдзень

Прыклад 19.3. Каментарыі ў праграме.

```
#жоўты круг
turtle.penup()
turtle.setpos(0,0)
turtle.pendown()
turtle.colormode(255)
turtle.color(255,165,0)
turtle.fillcolor(255,255,0)
turtle.begin_fill()
#радыус круга 50
turtle.circle(50)
turtle.end_fill()
```

Тут каментарыямі з'яўляюцца два радкі.

```
#жоўты круг
#радыус круга 50
```

Першы радок тлумачыць, што ў праграме ёсць каманды для малявання круга жоўтага колеру. Другі каментарый тлумачыць прызначэнне каманды, што ідзе за ім.

Каманда	Дзеянне
`circle(r)`	Маляваць акружнасць радыусам `r`
`setpos(x, y)`	Перамясціць *Чарапаху* ў пункт з каардынатамі (`x`, `y`)
`setheading(x)`	Задаць напрамак руху *Чарапахі*
`setup(w, h)`	Змяніць памеры акна *Чарапахі*: `w` — шырыня акна, `h` — вышыня акна
`towards(x, y)`	Атрымаць вугал паміж бягучым напрамкам *Чарапахі* і прамой ад *Чарапахі* да пункта (`x`, `y`)
`distance(x, y)`	Атрымаць адлегласць да пункта (`x`, `y`)

Напрамак *Чарапахі* ў камандзе `setheading(x)` задаецца велічынёй вугла. Некаторыя значэнні параметра `x` прыведзены ў прыкладзе 19.2.

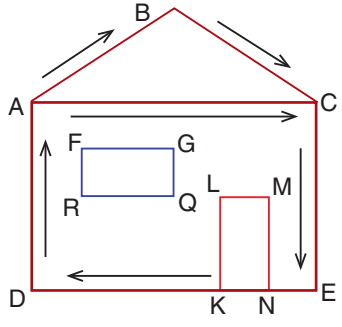
Калі тэкст праграмы вялікі, то яе складана чытаць, таму ў праграме часта пішуць каментарыі — радкі тэксту, што тлумачаць напісаныя каманды. Перад тэкстам каментарыяў ставіцца знак # (прыклад 19.3). Каментарыі можна пісаць па-беларуску. Выканаўца прапускае іх пры выкананні праграмы.

19.2. Атрыманне відарыса доміка

Прыклад 19.4. Складзі алгарытм пабудовы відарыса доміка і напішаць праграму для выканаўцы *Чарапаха*.

Выберам наступны алгарытм пабудовы відарыса:

1. Пабудаваць адрэзкі для даху: AB, BC, CA.
2. Пабудаваць адрэзкі для дома: AD, DE, EC.
3. Пабудаваць адрэзкі для дзвярэй: KL, LM, MN.
4. Пабудаваць адрэзкі для акна: FG, GQ, QR, RF.



Для падбору памераў і каардынат пажадана спачатку намаляваць выяву на аркушы паперы ў клетку.

Слоўнае апісанне алгарытму:

Падняць пяро
У пункт (-150,50)
Апусціць пяро
Пабудаваць дах

Прыклад 19.4. Праграма малявання доміка.

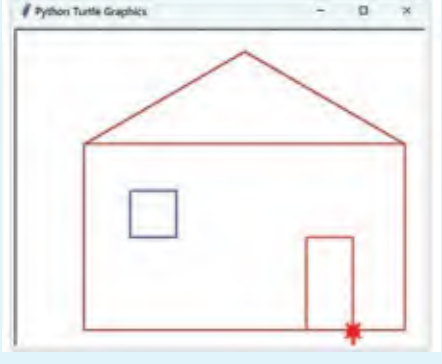
```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
turtle.penup()
turtle.setpos(-150,50)
turtle.pendown()
turtle.color('brown')
#дах
turtle.left(30)
turtle.forward(200)
turtle.right(60)
turtle.forward(200)
turtle.right(150)
turtle.forward(346)
#дом
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(346)
turtle.left(90)
turtle.forward(200)
#акно
turtle.penup()
turtle.setpos(-100,0)
turtle.pendown()
turtle.setheading(0)
turtle.color('blue')
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
```

Прыклад 19.4. *Працяг.*

```
#дзверы
turtle.penup()
turtle.setpos(90,-150)
turtle.pendown()
turtle.setheading(90)
turtle.color('red')
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(100)
```

Вынік работы праграмы

Прыклад 19.5. Зменены малюнак.




```
    Налева (30)
    Наперад (200)
    Направа (60)
    Наперад (200)
    Направа (150)
    Наперад (346)
  Пабудаваць дом
    Налева (90)
    Наперад (200)
    Налева (90)
    Наперад (346)
    Налева (90)
    Наперад (200)
  Падняць пяро
  У пукнт (-100,0)
  Апусціць пяро
  Пабудаваць акно
  Падняць пяро
  У пункт (90,-150)
  Апусціць пяро
  Пабудаваць  дзверы.
```

Напісаныя праграмы з некаторымі змяненнямі можна выкарыстоўваць для рашэння іншых задач.

Прыклад 19.5. Змяніць відарыс доміка з прыкладу 19.4.

Пры параўнанні новага відарыса і відарыса з прыкладу 19.4 выяўляюцца два адрозненні: першае — у памерах акна *Чарапахі*, другое — у форме акна доміка. У цэлым малюнкі падобныя. Значыць, для стварэння відарыса новага доміка можна выкарыстоўваць праграму

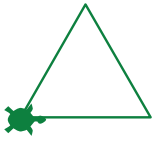
з прыкладу 19.4. Унясём змяненні ў тэкст праграмы ў акне рэдактара асяроддзя праграмірання, выкарыстаўшы правілы рэдагавання тэксту:

- дададзім каманду змянення акна ў пачатак праграмы;
- зменім фрагмент малявання акна доміка.

Астатнія каманды ў праграме застануцца без змяненняў.

19.3. Атрыманне відарыса трохвугольнікаў

Прыклад 19.6. Напісаць праграму пабудовы трохвугольніка.



У трохвугольніка тры стараны аднолькавай даўжыні. Усе вуглы роўны па 60°. Вугал павароту *Чарапахі* роўны 180° − 60° = 120°.

Слоўнае апісанне алгарытму 1 пабудовы трохвугольніка:

Наперад (100)
Налева (120)
Наперад (100)
Налева (120)
Наперад (100).

Аналагічны вынік можна атрымаць і з дапамогай іншага алгарытму (алгарытм 2).

Прыклад 19.5. *Працяг.* Змяненні ў праграме.

```
import turtle
turtle.shape('turtle')
turtle.setup(450,350)
#акно
turtle.penup()
turtle.setpos(-100,0)
turtle.pendown()
turtle.setheading(0)
turtle.color('blue')
turtle.forward(100)50
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(100)50
turtle.right(90)
turtle.forward(50)
```

Чырвонай рамкай абведзены радкі, якія адлюстроўваюць неабходныя змяненні. Замест закрэсленых лікаў трэба запісаць лікі, што стаяць пасля дужак.

Прыклад 19.6. Праграма пабудовы трохвугольніка.

Алгарытм 1

```
import turtle
turtle.shape('turtle')
turtle.color('green')
turtle.pensize(2)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
```

Прыклад 19.7. Праграма пабудовы трохвугольніка.

Алгарытм 2

```
import turtle
turtle.shape('turtle')
turtle.color('green')
turtle.pensize(2)
turtle.setpos(100,0)
turtle.setpos(50,75)
turtle.setpos(0,0)
```

Прыклад 19.8. Праграма малявання елкі.

```
import turtle
turtle.shape('turtle')
turtle.color('green')
turtle.pensize(2)
#ніжні трохвугольнік
turtle.penup()
turtle.setpos(-50,-85)
turtle.pendown()
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
#сярэдні трохвугольнік
turtle.penup()
turtle.setpos(-50,0)
turtle.pendown()
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
```

У пункт (100,0)
У пункт (50,75)
У пункт (0,0).

Праграма гэтага алгарытму прыведзена ў прыкладзе 19.7.

Калі параўнаць два алгарытмы, то можна зрабіць наступныя высновы:

1. Другі алгарытм мае меншую колькасць каманд, чым першы.

2. Трохвугольнік, пабудаваны па алгарытме 1, можа размяшчацца ў любым месцы плоскасці. Яго месцазнаходжанне залежыць толькі ад пачатковага становішча *Чарапахі*.

3. Трохвугольнік, пабудаваны па алгарытме 2, можа быць намаляваны толькі ў адным месцы плоскасці. Яго месцазнаходжанне вызначана каардынатамі вяршынь.

4. Першы алгарытм, нягледзячы на большую колькасць каманд, з'яўляецца больш універсальным. Яго можна выкарыстоўваць для пабудовы іншых відарысаў.

Прыклад 19.8. Напісаць праграму пабудовы елкі, якая складаецца з трох аднолькавых трохвугольнікаў.

Трохвугольнікі пабудуем, выкарыстаўшы алгарытм 1. Алгарытм пабудовы елкі можа быць наступным:

```
Падняць пяро
У пункт (-50,-85)
Апусціць пяро
Пабудаваць трохвугольнік
Падняць пяро
У пункт (-50,0)
Апусціць пяро
Пабудаваць трохвугольнік
Падняць пяро
У пункт (-50,85)
Апусціць пяро
Пабудаваць трохвугольнік.
```

Для пабудовы трохвугольніка ў праграме трэба тры разы скапіраваць фрагмент праграмы з прыкладу 19.6.

Прыклад 19.9. Змяніць праграму з прыкладу 19.6 для атрымання прамавугольнага трохвугольніка з вугламі 45°, 45°, 90° і даўжынёй катэтаў 100.

Вызначым адрозненні дадзенага трохвугольніка ад трохвугольніка ў прыкладзе 19.6.

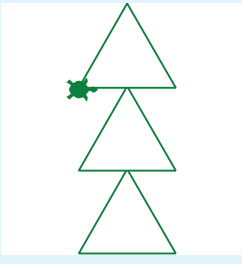
1. У гэтага трохвугольніка вуглы неаднолькавыя. *Чарапаха* павінна двойчы павярнуцца на вугал 135° (180° – 45°) і затым на вугал 90°.

2. Даўжыні дзвюх старон дадзенага трохвугольніка нам вядомы,

Прыклад 19.8. *Працяг.*

```
#верхні трохвугольнік
turtle.penup()
turtle.setpos(-50,85)
turtle.pendown()
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
```

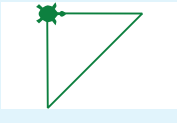
Вынік работы праграмы



Прыклад 19.9. Праграма пабудовы прамавугольнага трохвугольніка.

```
import turtle
turtle.shape('turtle')
turtle.color('green')
turtle.pensize(2)
turtle.forward(100)
turtle.right(135)
turtle.forward(141)
turtle.right(135)
turtle.forward(100)
turtle.right(90)
```

Вынік работы праграмы



Для вылічэння адлегласці можна выкарыстоўваць каманду `distance`. Для гэтага трэба ў праграме замяніць каманду

```
turtle.forward(141)
```

на каманду

```
turtle.forward
(turtle.distance(0,100))
```

Аднак пры такім падыходзе мы страцім універсальнасць малявання нашага трохвугольніка: яго месцазнаходжанне будзе прывязана да пункта (0, 100).

Прыклад 19.10. Праграма атрымання відарыса лічбы «3».

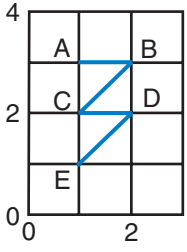
```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
turtle.penup()
turtle.setpos(-50,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
```

Вынік работы праграмы

а трэцяй — не. Для вылічэння яе даўжыні можна скарыстацца тым, што гіпатэнуза ў такога трохвугольніка прыкладна ў 1,41[1] раза большая за катэт.

19.4. Атрыманне відарыса лічбы «3»

Прыклад 19.10. Напісаць праграму пабудовы відарыса лічбы «3» у паштовым індэксе з дапамогай выканаўцы *Чарапаха*.



Алгарытм пабудовы відарыса:

У пункт A(-50,60)
Апусціць пяро
Пабудаваць лічбу, рухаючыся па адрэзках AB, BC, CD, DE.

Дадзеная выява складаецца з двух аднолькавых фрагментаў, кожны з якіх уяўляе сабой прамавугольны трохвугольнік без аднаго катэта. Таму можна карыстацца праграмай з прыкладу 19.9.

Даўжыня гарызантальнай лініі роўна 30, дыяганальнай — $30 \times 1{,}41 \approx 42$.

[1] Чаму гэта роўнасць правільная, вы даведаецеся на ўроках матэматыкі.

У некаторых відарысах часта паўтараюцца аднолькавыя фрагменты. Для стварэння праграм пабудовы такіх малюнкаў можна капіраваць фрагмент праграмы і выкарыстоўваць яго патрэбную колькасць разоў так, як гэта зроблена ў прыкладзе 19.8 для пабудовы елкі.

Прыклад 19.11. Напісаць праграму для пабудовы трох лічбаў «3» аднолькавага памеру, намаляваных чырвоным, зялёным і сінім колерамі.

Мы бачым, што праграму пабудовы гэтага відарыса можна скласці на аснове праграмы з прыкладу 19.8. Відарыс першай лічбы пачынаецца ад верхняга пункта злева, яго каардынаты (–50, 60). Каардынаты такога ж пункта для другой лічбы (0, 60), для трэцяй — (50, 60).

Такім чынам, для стварэння відарыса з трох лічбаў «3» трэба скапіраваць у тэксце праграмы з прыкладу 19.10 фрагмент

```
turtle.penup()
turtle.setpos(-50,60)
turtle.pendown()
turtle.forward(30)
```

Значэнні параметраў каманд *Чарапахі* можна запісваць выразам. Напрыклад, замест каманды `turtle.forward(42)` можна запісаць каманду `turtle.forward(30*1.41)`.

Прыклад 19.11. Праграма пабудовы відарыса трох аднолькавых лічбаў рознага колеру.

```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
#першая тройка
turtle.penup()
turtle.color('red')
turtle.setpos(-50,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
#другая тройка
turtle.penup()
turtle.color('green')
turtle.setpos(0,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
```

Прыклад 19.11. *Працяг.*

```
#трэцяя тройка
turtle.penup()
turtle.color('blue')
turtle.setpos(50,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
```

Вынік работы праграмы

```
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
```

затым трэба ўставіць скапіраваны фрагмент патрэбную колькасць разоў. У кожным з устаўленых фрагментаў змяніць параметры каманды `turtle.setpos(-50,60)` — задаць каардынаты пачатковых пунктаў. Таксама ў кожным фрагменце трэба дапісаць каманды змянення колеру.

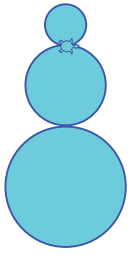
1. Як атрымаць даведачную інфармацыю аб камандах `setheading`, `pensize`?
2. Як задаць колер лініі *Чарапахі*?
3. Як зафарбаваць намаляваную *Чарапахай* фігуру?
4. Якое значэнне павінна быць у каманды `forward`, каб *Чарапаха* перамясцілася па дыяганалі квадрата з даўжынёй стараны 120?

Практыкаванні

1 Змяніце праграму з прыкладу 19.1 для атрымання відарысаў.

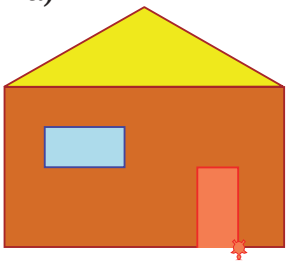
а)

б)

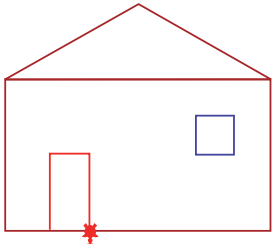


2 Змяніце праграму з прыкладу 19.4 для атрымання відарысаў. Канечнае месцазнаходжанне *Чарапахі* можа быць іншым.

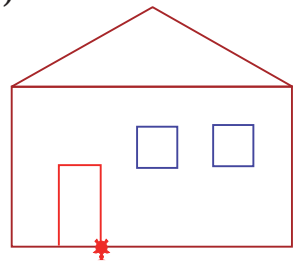
а)



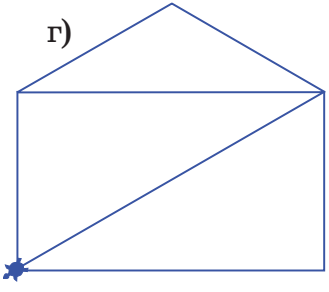
б)



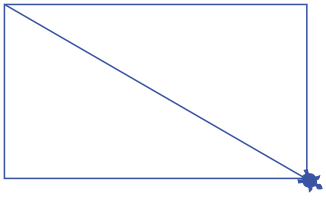
в)



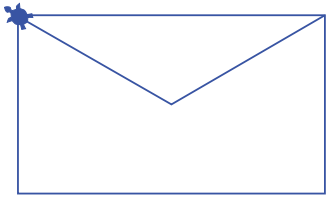
г)



д)

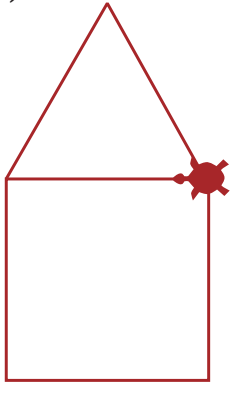


е)

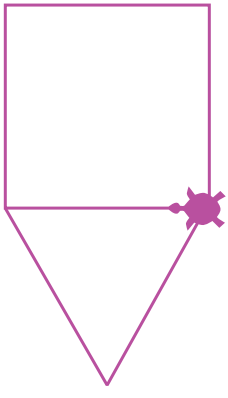


3 Выкарыстайце праграмы з прыкладаў 19.6 і 19.8 для атрымання відарысаў.

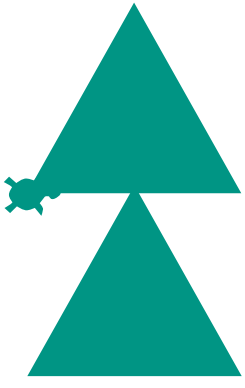
а)



б)



в)



4 Устаўце ў рэдактар асяроддзя праграмірaвання IDLE тэкст праграмы, вызначыце вынік яе выканання.

```
import turtle
turtle.shape('turtle')
#літара Б
turtle.left(90)
turtle.penup()
turtle.forward(30)
turtle.pendown()
turtle.right(90)
turtle.forward(30)
turtle.right(90)
turtle.forward(30)
turtle.right(90)
turtle.forward(30)
turtle.right(90)
turtle.forward(60)
turtle.right(90)
turtle.forward(30)
#пераход
turtle.penup()
turtle.setpos(40, 0)
turtle.pendown()
```

```
#літара Г
turtle.left(90)
turtle.forward(60)
turtle.right(90)
turtle.forward(30)
#пераход
turtle.penup()
turtle.setpos(80, 0)
turtle.pendown()
#літара У
turtle.forward(30)
turtle.left(90)
turtle.forward(60)
turtle.left(90)
turtle.penup()
turtle.forward(30)
turtle.pendown()
turtle.left(90)
turtle.forward(30)
turtle.left(90)
turtle.forward(30)
```

Змяніце праграму для пабудовы відарысаў.

а) 

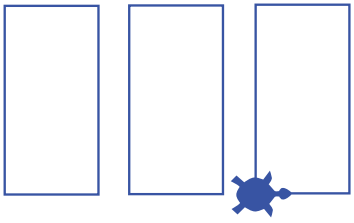
б) 

в) У праграму пабудовы відарыса з пункта а) унясіце змяненні так, каб замест літары Б атрымлівалася літара Л.

5 Скапіруйце і ўстаўце ў рэдактар асяроддзя праграмірования IDLE тэкст праграмы, вызначце вынік яе выканання.

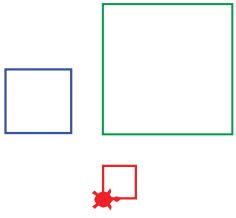
```
import turtle
turtle.shape('turtle')
turtle.color('blue')
turtle.pensize(2)
turtle.forward(60)
turtle.left(90)
turtle.forward(120)
turtle.left(90)
turtle.forward(60)
turtle.left(90)
turtle.forward(120)
turtle.left(90)
```

Змяніце праграму для пабудовы відарыса.

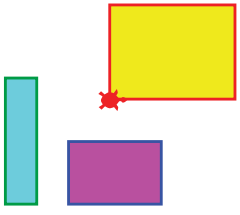


6* Складзіце праграмы для атрымання відарысаў а)—г). Падумайце, якія з ужо выкананых практыкаванняў можна выкарыстаць для атрымання гэтых відарысаў.

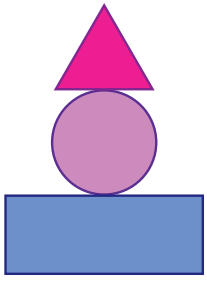
а)



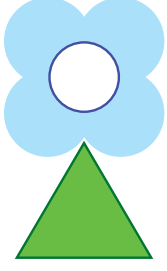
б)



в)



г)



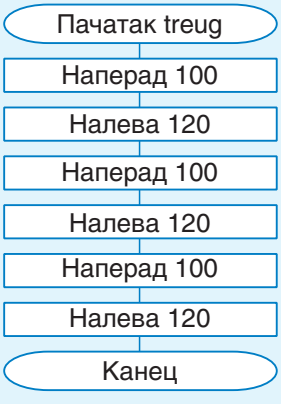
§ 20. Складанне праграм. Выкарыстанне падпраграм (дапаможных алгарытмаў)

На заранку стварэння першых камп'ютараў пры распрацоўцы праграм ужываўся прыём праекціравання «знізу ўверх»: спачатку стваралі найбольш простыя падпраграмы, затым іх выкарыстоўвалі ў больш складаных праграмах. У сярэдзіне 60-х гг. XX ст. стаў прымяняцца метад пакрокавай дэталізацыі алгарытмаў (праекціраванне «зверху ўніз»). Гэты метад закладзены ў аснову працэдурных моў праграмірання (Pascal, C++, Python і інш.).

Прыклад 20.1. Маляванне елкі.

Блок-схема алгарытму малявання елкі ўключае:

1. Блок-схему дапаможнага алгарытму для малявання аднаго элемента елкі (трохвугольніка).

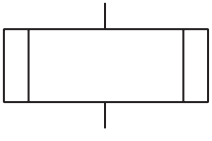


20.1. Паняцце дапаможнага алгарытму

Як вядома з папярэдняга параграфа, выканаўцу *Чарапаха* нярэдка даводзіцца будаваць адзін і той жа відарыс у адной праграме некалькі разоў. Пабудову такога відарыса зручна афарміць у выглядзе асобнага алгарытму. Такія алгарытмы называюць дапаможнымі.

Дапаможны алгарытм — алгарытм, які можна цалкам выкарыстоўваць у іншых алгарытмах.

Дапаможны алгарытм можна выкарыстоўваць неабходную колькасць разоў, звяртаючыся да яго назвы (імя). Для звароту да дапаможнага алгарытму ў блок-схемах выкарыстоўваецца блок.



Дапаможны алгарытм у мове Python запісваецца ў выглядзе функцыі.

```
def імя_функцыі():
  каманда1
  каманда2
...
```

Загаловак функцыі

Каманды (цела функцыі)

Імя функцыі можа змяшчаць літары лацінскага алфавіта, лічбы, знак падкрэслівання. Першы сімвал у імені працэдуры не можа быць лічбай. Пасля імя функцыі ў дужках могуць указвацца параметры, ад якіх залежыць вынік работы функцыі. Нават калі параметраў няма, наяўнасць дужак з'яўляецца абавязковай. Пасля дужак ставіцца двукроп'е. Усе каманды, якія адносяцца да цела функцыі, пішуцца са зрухам направа. Зрух задаюць клавішай Tab.

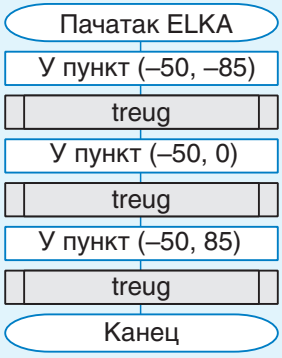
Каманду выканання дапаможнага алгарытму называюць **выклікам функцыі**. Каманду выкліку запісваюць у асноўным алгарытме (праграме) шляхам указання імя працэдуры.

Прыклад 20.1. Напісаць праграму для малявання елкі з выкарыстаннем дапаможнага алгарытму.

У прыкладзе 19.8 мы ўжо будавалі елку, выкарыстоўваючы відарыс трохвугольніка. Калі прааналізаваць алгарытм, то можна

Прыклад 20.1. *Працяг.*

2. Блок-схему пабудовы елкі з выкарыстаннем дапаможнага алгарытму treug.



Праграма пабудовы елкі

```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
def treug():
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
#ніжні трохвугольнік
turtle.penup()
turtle.setpos(-50, -85)
turtle.pendown()
treug()
#сярэдні трохвугольнік
turtle.penup()
turtle.setpos(-50, 0)
turtle.pendown()
treug()
#верхні трохвугольнік
turtle.penup()
turtle.setpos(-50, 85)
turtle.pendown()
treug()
```

Прыклад 20.2. Зменена праграма малявання елкі.

```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
def treug():
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
def p(x, y):
    turtle.penup()
    turtle.setpos(x,y)
    turtle.pendown()
#ніжні трохвугольнік
p(-50, -85)
treug()
#сярэдні трохвугольнік
p(-50, 0)
treug()
#верхні трохвугольнік
p(-50, 85)
treug()
```

Калі параўнаць колькасць радкоў у праграмах з прыкладаў 19.8 і 20.2, то можна заўважыць, што іх стала менш. Выкарыстанне дапаможных алгарытмаў дазваляе зрабіць праграму больш кароткай.

Прыклад 20.3. Новыя змяненні ў праграме малявання елкі.

У целе функцыі `treug` выдалім апошнюю каманду `turtle.left(120)`, якая разгортвала *Чарапаху* ў пачатковае становішча. Замест яе ў функцыю `p` дададзім каманду `turtle.setheading(0)`, якая задае вугал павароту, роўны 0.

ўбачыць, што каманда **Пабудаваць трохвугольнік** запісана тройчы. Гэта значыць, што можна не капіраваць каманды для малявання трохвугольніка тры разы, а аформіць дапаможны алгарытм, які будзе будаваць трохвугольнік.

Трохвугольнік будаваўся з дапамогай наступных каманд:

```
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120).
```

Апішам гэту паслядоўнасць каманд у выглядзе дапаможнага алгарытму `treug`, які будзе выкарыстоўвацца ў нязменным выглядзе некалькі разоў (у дадзеным выпадку тройчы).

У праграме з прыкладу 19.8 ёсць яшчэ тры каманды:

```
turtle.penup()
turtle.setpos(…, …)
turtle.pendown().
```

Гэтыя каманды дазваляюць ажыццявіць пераход ад аднаго трохвугольніка да іншага. Адрозненне ў выкарыстанні гэтых каманд толькі ў каардынатах, у якія павінна перамясціцца *Чарапаха*. Можна аформіць яшчэ

адну функцыю, якая будзе ажыццяўляць пераход. Гэта функцыя будзе залежаць ад кардынат пункта, у які трэба перамясціць *Чарапаху*.

Назавём гэту функцыю p(x, y). У прыкладзе 20.2 прыведзена зменена праграма.

Калі лічыць, што маляванне трохвугольніка залежыць ад пункта, з якога *Чарапаха* пачынае яго маляваць, то функцыя малявання трохвугольніка таксама будзе залежаць ад параметраў (x, y). Алгарытм малявання трохвугольніка ў гэтым выпадку будзе пачынацца з каманды перамяшчэння *Чарапахі* ў пункт з каардынатамі (x, y). Праграма паказана ў прыкладзе 20.3.

У дадзеным прыкладзе функцыя p выклікаецца з функцыі treug.

20.2. Рашэнне практычных задач з выкарыстаннем функцый

Прыклад 20.4. Аформіць з дапамогай функцый праграму малявання трох троек на паштовым канверце з прыкладу 19.11.

Функцыя для малявання адной тройкі cifr_3 будзе функцыяй, якая залежыць ад пачатковага месцазнаходжання *Чарапахі* —

Прыклад 20.3. *Працяг.* Новыя змяненні ў праграме малявання елкі.

```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
def treug(x, y):
    p(x, y)
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
    turtle.left(120)
    turtle.forward(100)
def p(x, y):
    turtle.penup()
    turtle.setpos(x, y)
    turtle.pendown()
    turtle.setheading(0)
#ніжні трохвугольнік
treug(-50, -85)
#сярэдні трохвугольнік
treug(-50, 0)
#верхні трохвугольнік
treug(-50, 85)
```

Прыклад 20.4. Праграма пабудовы відарыса з прыкладу 19.11.

```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
def cifr_3(x, y, c):
    p(x, y)
    turtle.color(c)
    turtle.forward(30)
    turtle.right(135)
    turtle.forward(42)
    turtle.left(135)
    turtle.forward(30)
    turtle.right(135)
```

Прыклад 20.4. *Працяг.*

```
    turtle.forward(42)
def p(x, y):
    turtle.penup()
    turtle.setpos(x, y)
    turtle.pendown()
    turtle.setheading(0)
#першая тройка
cifr_3(-50, 60, 'red')
#другая тройка
cifr_3(0, 60, 'green')
#трэцяя тройка
cifr_3(50, 60, 'blue')
```

Прыклад 20.5. Праграма пабудовы відарыса.

```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
def cifr_1(x, y, c):
    p(x, y)
    turtle.color(c)
    turtle.left(45)
    turtle.forward(42)
    turtle.right(135)
    turtle.forward(60)
def cifr_3(x, y, c):
    p(x, y)
    turtle.color(c)
    turtle.forward(30)
    turtle.right(135)
    turtle.forward(42)
    turtle.left(135)
    turtle.forward(30)
    turtle.right(135)
```

каардынат (x, y). Функцыю p(x, y), якая перамяшчае *Чарапаху* ў патрэбны пункт, возьмем з прыкладу малявання елкі. Паколькі кожная тройка павінна малявацца сваім колерам, дададзім параметр c у апісанне функцыі.

На прыкладзе малявання елкі і трох троек мы ўбачылі, што дапаможныя алгарытмы дазваляюць скарачаць код праграмы, паколькі пазбаўляюць ад неабходнасці запісваць некалькі разоў аднолькавыя часткі праграмы.

Дапаможныя алгарытмы выкарыстоўваюць і тады, калі задача разбіваецца на часткі — падзадачы. Дапаможны алгарытм рашае некаторую падзадачу асноўнай задачы.

Прыклад 20.5. Напісаць праграму для пабудовы відарыса, які складаецца з лічбаў «1», «3» і «7», намаляваных чырвоным, зялёным і сінім колерамі.

У дадзеным выпадку ў відарысе няма фрагментаў, якія паўтараюцца. Аднак задача разбіваецца на тры падзадачы:

1. Намаляваць лічбу «1».
2. Намаляваць лічбу «3».
3. Намаляваць лічбу «7».

Функцыю для малявання лічбы «3» можна ўзяць з папярэдня-

га прыкладу. Застаецца напісаць функцыі для малявання лічбаў «1» і «7».

Над рашэннем рэальных задач праекта могуць працаваць некалькі чалавек. Кожны выконвае сваю частку работы і афармляе яе як асобны дапаможны алгарытм — функцыю. Важна, каб функцыі, напісаныя рознымі людзьмі, правільна выконваліся ў адным праекце. Для гэтага ўстанаўліваюцца адзіныя падыходы да напісання кода праграмы.

Для выканаўцы *Чарапаха* ўстановім наступнае правіла: маляванне любой фігуры пачынаецца з перамяшчэння *Чарапахі* ў пачатковы пункт. Пачатковым пунктам дамовімся лічыць ніжні левы вугал відарыса. У пачатковым пункце *Чарапаха* глядзіць направа (на ўсход).

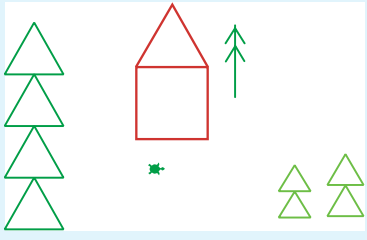
Разгледзім наступны прыклад. Няхай некалькі шасцікласнікаў атрымалі заданне распрацаваць праграму малявання некаторага «пейзажа». «Пейзаж» складаецца з наступных элементаў: дом, тры елкі (адна вялікая і дзве маленькія) і сасна (прыклад 20.6).

Спісак функцый для малявання «пейзажа» паказаны ў прыкладзе 20.7

Прыклад 20.5. *Працяг.*

```
    turtle.forward(42)
def cifr_7(x, y, c):
    p(x, y)
    turtle.color(c)
    turtle.forward(30)
    turtle.right(135)
    turtle.forward(42)
    turtle.left(45)
    turtle.forward(30)
def p(x,y):
    turtle.penup()
    turtle.setpos(x,y)
    turtle.pendown()
    turtle.setheading(0)
cifr_1(0, 30, 'red')
cifr_3(50, 60, 'green')
cifr_7(100, 60, 'blue')
p(150,0)
```

Прыклад 20.6. «Пейзаж».



Прыклад 20.7. Спісак функцый для малявання «пейзажа».

- `treug` — пабудова трохвугольніка;
- `p` — перамяшчэнне *Чарапахі* ў пачатковы пункт;
- `b_el` — маляванне вялікай елкі;
- `m_el` — маляванне маленькай елкі;
- `dom` — маляванне дома;
- `sosna` — маляванне сасны.

Прыклад 20.8. Праграма для малявання «пейзажа».

```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
def treug(x, y, d):
    p(x,y)
    turtle.forward(d)
    turtle.left(120)
    turtle.forward(d)
    turtle.left(120)
    turtle.forward(d)
def p(x,y):
    turtle.penup()
    turtle.setpos(x,y)
    turtle.pendown()
    turtle.setheading(0)
def m_el(x, y, d):
    treug(x, y, d)
    treug(x, y + d * 0.86, d)
def b_el(x, y, d):
    m_el(x, y, d)
    m_el(x, y + d * 1.72, d)
def dom(x, y, d):
    p(x,y)
    turtle.forward(d)
    turtle.left(90)
    turtle.forward(d)
    turtle.left(60)
    treug(x, y + d, d)
    turtle.left(30)
    turtle.forward(d)
def sosna(x, y, d):
    p(x + d / 8 , y)
    turtle.left(90)
    turtle.forward(d)
    p(x, y + d / 2)
    turtle.left(60)
    turtle.forward(d / 4)
    turtle.right(120)
    turtle.forward(d / 4)
    p(x, y + 3 * d / 4)
    turtle.left(60)
    turtle.forward(d / 4)
    turtle.right(120)
    turtle.forward(d / 4)
```

Шасцёра шасцікласнікаў могуць размеркаваць работу паміж сабой наступным чынам:

- першы піша дапаможны алгарытм малявання сасны;
- другі піша дапаможны алгарытм малявання дома;
- трэці піша дапаможны алгарытм малявання трохвугольніка. У гэтым алгарытме пажадана дадаць новы параметр — даўжыня стараны трохвугольніка;
- чацвёрты піша дапаможны алгарытм для маленькай елкі, узяўшы за аснову дапаможны алгарытм малявання трохвугольніка. Каардынаты x, y абодвух трохвугольнікаў аднолькавыя, каардынату y можна вылічыць па формуле: $y + 0{,}86 \cdot d$, дзе d — даўжыня стараны трохвугольніка;
- пяты піша дапаможны алгарытм для вялікай елкі, узяўшы за аснову дапаможны алгарытм малявання маленькай елкі;
- шосты піша асноўны алгарытм, размяшчаючы элементы «пейзажа» на полі *Чарапахі*.

Праграма малявання «пейзажа» паказана ў прыкладзе 20.8.

Маючы ў сваім распараджэнні дапаможныя алгарытмы, можна лёгка атрымліваць іншыя «пей-

зажы». Пры гэтым не спатрэбіцца перапісваць дапаможныя алгарытмы. Будзе дастаткова выбраць месца размяшчэння аб'екта на полі *Чарапахі*, пазначыўшы каардынаты пачатковага пункта аб'екта.

Напрыклад, можна захаваць усе функцыі з прыкладу 20.8, а ў тэкст асноўнай часткі праграмы ўнесці змяненні (прыклад 20.9).

Вынік работы праграмы пасля змянення паказаны ў прыкладзе 20.10.

Прыклады, разгледжаныя ў гэтым параграфе, наглядна дэманструюць неабходнасць выкарыстання дапаможных алгарытмаў. Яны дазваляюць рашаць складаныя задачы больш эфектыўным і зручным спосабам, палягчаюць і паскараюць распрацоўку праграмнага кода, павялічваюць яго чытэльнасць. Выкарыстанне функцый апраўданае, калі:

- для атрымання рашэння задачы некаторыя дзеянні даводзіцца паўтараць неаднаразова (як у прыкладах 20.2, 20.4);
- задача разбіваецца на незалежныя падзадачы (як у прыкладах 20.5, 20.8, 20.9).

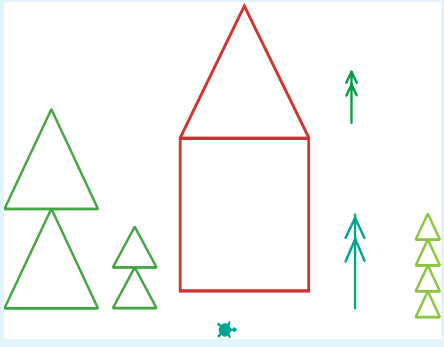
Прыклад 20.8. *Працяг.*

```
turtle.color('green')
b_el(-250,-200, 100)
turtle.color('lime')
m_el(210, -180, 50)
m_el(290, -180, 60)
turtle.color('brown')
dom(-30, -50, 120)
turtle.color('darkgreen')
sosna(120, 20, 120)
p(0,-100)
```

Прыклад 20.9. Змяненні ў праграме.

```
turtle.color('green')
m_el(-350, -120, 150)
m_el(-170, -120, 60)
turtle.color('lime')
b_el(300,-135, 40)
turtle.color('brown')
dom(-70, -100, 200)
turtle.color('darkgreen')
sosna(190, 120, 70)
turtle.color('teal')
sosna(190, -120, 120)
p(0,-150)
```

Прыклад 20.10. Вынік змяненняў.



1. Якія алгарытмы называюцца дапаможнымі?
2. Для чаго патрэбны дапаможныя алгарытмы?
3. Якое ключавое слова выкарыстоўваецца для апісання функцыі?
4. Як выконваецца выклік функцыі?
5. Як зразумець, якія каманды ўтвараюць цела функцыі?

Практыкаванні

1 Змяніце праграму малявання елкі так, каб трохвугольнік маляваўся зафарбаваным.

2 Выкарыстаўшы складзеныя раней праграмы для выканаўцы *Чарапаха* як дапаможныя, складзіце праграму для пабудовы відарыса.

3 Выкарыстаўшы функцыі з прыкладу 20.8, прыдумайце свой «пейзаж».

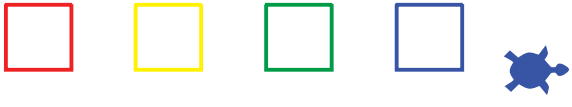
4 Складзіце праграмы пабудовы відарыса ўсіх лічбаў ад 0 да 9.

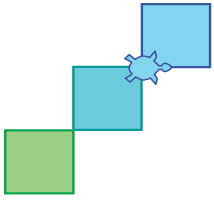
5 Запішыце праграмы напісання слоў выканаўцам *Чарапаха*. Выкарыстайце дапаможныя алгарытмы малявання літар.

а) 

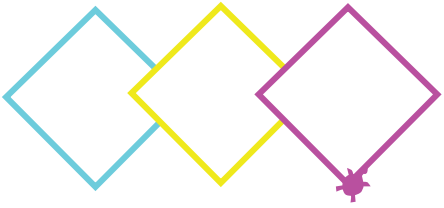
б) 

6 Складзіце праграмы пабудовы відарысаў. Што могуць азначаць гэтыя відарысы?

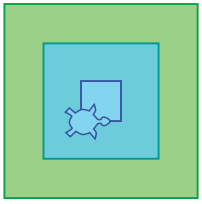
а) 

б) 

в)



г)



7 Складзіце праграму для пабудовы відарыса.