Глава 6
АЛГАРЫТМЫ І ВЫКАНАЎЦЫ

§ 16. Паняцце алгарытму і выканаўцы

У III ст. да н. э. старажытнагрэчаскі матэматык Эўклід сфармуляваў правіла вылічэння найбольшага агульнага дзельніка двух натуральных лікаў. Гэта правіла лічаць першым алгарытмам.

Эўклід

Аль-Харэзмі

Тэрмін *алгарытм* (лац. *algorithmus*) паходзіць ад імя арабскага матэматыка Мухамеда аль-Харэзмі (787–850). Ён распрацаваў правілы выканання чатырох арыфметычных дзеянняў, якія прымяняюцца і сёння.

Прыклад 16.1. Падключэнне да сеткі Інтэрнэт праз Wi-Fi ў грамадскім месцы.

1. Уключыць Wi-Fi на мабільным устройстве.
2. Выканаць пошук даступных сетак.
3. Выбраць даступную сетку.
4. Калі сетка мае ключ доступу, удакладніць яго ў адміністратара.
5. Увесці ключ доступу.

Дадзены алгарытм складаецца з пяці каманд.

16.1. Паняцце алгарытму

У паўсядзённым жыцці нам даводзіцца рашаць шмат задач, простых і складаных: збор у школу, пакупка марожанага, званок па мабільным тэлефоне і г. д. Больш складана атрымаць адзнаку 10 па інфарматыцы, перамагчы на спаборніцтвах і інш.

Для рашэння любой задачы неабходна выканаць пэўныя дзеянні (прыклад 16.1).

Алгарытм — зразумелая і канечная паслядоўнасць дакладных дзеянняў (каманд), фармальнае выкананне якіх дазваляе атрымаць рашэнне пастаўленай задачы.

Каманда ў алгарытме — указанне на выкананне канкрэтнага дзеяння.

Для рашэння адной і той жа задачы могуць выкарыстоўвацца розныя алгарытмы. Напрыклад, для напісання віншавальнай паштоўкі адзін вучань можа выкарыстаць паперу і каляровыя алоўкі, другі — тэкставы рэдак-

тар, трэці — графічны рэдактар (прыклад 16.2).

16.2. Паняцце выканаўцы алгарытму

Кожны алгарытм ствараецца чалавекам ці групай людзей. Алгарытм выконваецца выканаўцамі алгарытмаў.

Выканаўца алгарытму — чалавек (група людзей) або тэхнічнае ўстройства, якія разумеюць каманды алгарытму і ўмеюць правільна іх выконваць.

Выканаўцам алгарытмаў з прыкладаў 16.1 і 16.2 можа быць чалавек. Выконваць алгарытм можа робат, дзіцячая цацка, экшн-камера, станкі з лікавым праграмным кіраваннем (ЛПК) (прыклад 16.3) і г. д.

Каманды, якія разумее і можа выканаць выканаўца, утвараюць **сістэму каманд выканаўцы.** У прыкладзе 16.4 прыведзена сістэма каманд выканаўцы *Робат-пыласос.* У залежнасці ад плошчы і асаблівасцей памяшкання чалавек можа задаваць розныя рэжымы працы (алгарытмы) *Робата-пыласоса.*

Алгарытмы, прызначаныя для выканання на камп'ютары, запісваюць на мове праграмірования.

Прыклад 16.2. Напісанне віншавальнай паштоўкі.

1. Адкрыць графічны рэдактар Paint.
2. Намаляваць паштоўку.
3. Раздрукаваць паштоўку.

Дадзены алгарытм складаецца з трох каманд.

Прыклад 16.3. Станкі з ЛПК, якія вырабляюцца ў Рэспубліцы Беларусь.

Прыклад 16.4. Сістэма каманд выканаўцы *Робат-пыласос.*

- Заніраваць памяшканне;
- рэгуляваць падачу вады;
- распазнаваць прадметы;
- выконваць уборку;
- выконваць самаачыстку.

Робат-пыласос не можа выканаць каманду, якая не ўваходзіць у яго сістэму каманд, напрыклад: **патэлефанаваць.**

Прыклад 16.5. Узоры камп'ютарных выканаўцаў.

- *Чарцёжнік*, *Робат*, *Чарапаха* рэалізаваны для розных моў праграмiравання;
- *Рыжы кот* рэалізаваны ў праграме Scratch.

Прыклад 16.6. Асяроддзем існавання выканаўцы алгарытму 16.1 можа быць толькі асяроддзе, у якім выкарыстоўваюцца мабільныя тэлефоны. Гэты алгарытм немагчыма выканаць пры адсутнасці сеткі і пункту доступу да сеткі Інтэрнэт.

Алгарытм з прыкладу 16.2 нельга выканаць без камп'ютара і прынтара.

Прыклад 16.7. Выкананне алгарытму выканаўцам *Шасцікласнік*.

Нумар каманды	Вынік выканання каманды
1	105
2	$105 \cdot 2 = 210$
3	$210 + 10 = 220$
4	$220 : 2 = 110$
5	$110 - 105 = 5$
6	5

Запіс алгарытму на мове праграмiравання называюць **праграмай.** Выканаўцам праграм з'яўляецца камп'ютар.

Камп'ютарны выканаўца — віртуальны аб'ект, які дзейнічае ў віртуальным асяроддзі (прыклад 16.5).

Для некаторых выканаўцаў патрабуюцца пэўныя ўмовы. Такія ўмовы называюць **асяроддзем існавання выканаўцы** (прыклад 16.6).

Выканаўца *Шасцікласнік* (асяроддзе існавання — 6-ы клас) умее:

- задумваць натуральны лік;
- выконваць арыфметычныя дзеянні над лікамі;
- запісваць лікі;
- знаходзіць найбольшы і найменшы лікі сярод дадзеных лікаў.

Яму прапануецца выканаць алгарытм:

1. Задумаць некаторы натуральны лік.
2. Памножыць задуманы лік на 2.
3. Да атрыманага здабытку дадаць 10.
4. Вынік падзяліць на 2.
5. Ад дзелі адняць задуманы лік.
6. Запісаць вынік.

(Разгледзьце прыклад 16.7.)

Няхай асяроддзе існавання выканаўцы *Пэндзаль* — аркуш паперы. Сістэма каманд выканаўцы *Пэндзаль*:

- стрэлка — выканаўца малюе адрэзак некаторай даўжыні ў напрамку, зададзеным стрэлкай;
- закрэсленая стрэлка — выканаўца рухаецца ў напрамку, зададзеным стрэлкай, не пакідаючы следу (прыклад 16.8).

Разгледзім алгарытм вызначэння сутачнай амплітуды тэмпературы паветра. Для гэтага трэба:

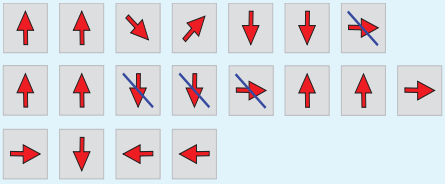
1. Вызначыць максімальную тэмпературу паветра за суткі.
2. Вызначыць мінімальную тэмпературу паветра за суткі.
3. Знайсці рознасць паміж максімальным і мінімальным значэннямі тэмператур (прыклад 16.9).

Выканаўцам гэтага алгарытму можа стаць любы чалавек, якому зразумелы каманды алгарытму.

Алгарытм вызначэння азімута на мясцовасці можа быць такім:

1. Сумясціць чырвоны канец магнітнай стрэлкі компаса з напрамкам на поўнач.

Прыклад 16.8. Выкананне алгарытму выканаўцы *Пэндзаль*.



Алгарытм

Вынік

Прыклад 16.9. Вызначэнне сутачнай амплітуды тэмпературы паветра выканаўцам *Шасікласнік* па табліцы.

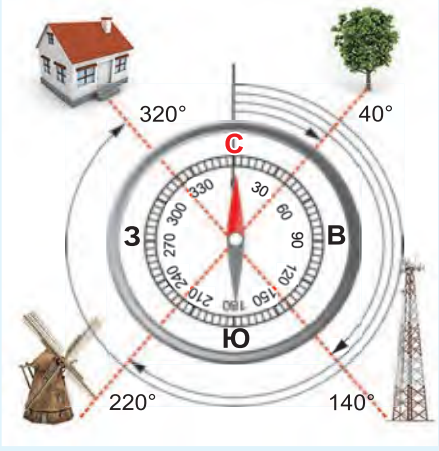
Час назірання	Тэмпература, °С
6.00	+10
12.00	+17
18.00	+14
24.00	+8

Максімальны вынік +17 °С, мінімальны — +8 °С. Амплітуда тэмператур паветра:

17 °С − 8 °С = 9 °С.

Вынік выканання алгарытму: 9.

Прыклад 16.10. Вызначэнне азімута для аб'ектаў на малюнку.



Даручым выкананне алгарытму выканаўцу *Шасцікласнік*, мяркуючы, што ён разумее і можа правільна выканаць каманды алгарытму.

Вынік выканання алгарытму: азімут на дрэва роўны 40°; азімут на млын роўны 220°; азімут на вежу сотавай сувязі роўны 140°.

2. Мысленна правесці прамую лінію ад цэнтра компаса да аб'екта.

3. Вызначыць вугал паміж стрэлкай на поўнач і ўяўнай лініяй да аб'екта ў напрамку гадзіннікавай стрэлкі (азімут на поўнач роўны 0°) (прыклад 16.10).

З курса матэматыкі вам вядомы алгарытм пабудовы прамавугольнай сістэмы каардынат:

1. Пабудаваць дзве перпендыкулярныя прамыя (гарызантальную і вертыкальную) і абазначыць: Ox і Oy.

2. Выбраць дадатны напрамак і адзначыць яго стрэлкай на кожнай прамой.

3. Адзначыць пачатак каардынат: пункт O (лік 0).

4. Адзначыць на кожнай прамой адзінкавыя адрэзкі.

1. Што такое алгарытм?
2. Што называецца камандай у алгарытме?
3. Што такое выканаўца алгарытму?
4. Хто можа быць выканаўцам алгарытму?
5. Што называюць сістэмай каманд выканаўцы?
6. Што такое асяроддзе існавання выканаўцы?

Практыкаванні

1 Прывядзіце прыклады алгарытмаў з паўсядзённага жыцця і вучэбнай дзейнасці.

2 Якія з наступных працэсаў можна апісаць у выглядзе алгарытмаў?

1. Замена кола ў аўтамабілі.
2. Напісанне сачынення.
3. Складанне двух дробаў.
4. Забіванне гола на футбольным матчы.
5. Атрыманне відарыса беларускага арнаменту, паказанага на малюнку справа.
6. Запіс шэрага ўсіх натуральных лікаў.

3* Рашыце метадам падбору задачу Аль-Харэзмі: «Я да трэці ліку дадаў адзінку і да чвэрці ліку дадаў адзінку. Памножыўшы гэтыя лікі, атрымаў 20. Які лік я ўзяў?»

4 Прывядзіце прыклады выканаўцаў алгарытмаў.

5 Напішыце сістэму каманд аднаго з выканаўцаў прыкладу 16.3.

6 Выканайце алгарытм з прыкладу 16.7 некалькі разоў для розных лікаў. Параўнайце атрыманыя вынікі. Зрабіце высновы.

7 Па камандах  ,  ,  ,  выканаўца *Пэндзаль* малюе частку акружнасці ў пазначаным напрамку. Вызначце вынік выканання наступнага алгарытму:

 .

8 Запішыце алгарытм марфалагічнага разбору прыметніка ў сказе. Выканайце гэты алгарытм для прыметніка «лічбавым» са сказу: «Сучасны чалавек жыве ў лічбавым свеце».

9* Прыдумайце выканаўцу алгарытмаў са сваёй сістэмай каманд і напішыце для яго алгарытм рашэння некаторай задачы.

§ 17. Спосабы запісу алгарытмаў

Прыклад 17.1. Алгарытм гульні «Магія лікаў» на дзіцячым прававым сайце[1].

1. Задумаць двухзначны лік.
2. Адняць ад задуманага ліку лічбы, якімі ён запісаны.
3. Знайсці знак атрыманай рознасці ў табліцы.
4. Уявіць гэты знак і круціць чароўнае кола.

Прыклад 17.2. Алгарытм прыгатавання беларускіх дранікаў.

1. Пачысціць бульбу.
2. Нацерці бульбу на дробнай тарцы.
3. Нацерці цыбуліну на дробнай тарцы.
4. Дадаць цыбулю ў бульбу.
5. Дадаць яйка, муку, соль і спецыі.
6. Усё добра размяшаць.
7. Разагрэць патэльню з алеем.
8. Выкласці лыжкай бульбяную масу на патэльню ў выглядзе аладкі і абсмажыць з двух бакоў да гатоўнасці.

Здаўна чалавек імкнуўся запісваць патрэбныя дзеянні ў кароткай і зразумелай форме. Так у розных сферах жыцця з'явіліся разнастайныя інструкцыі: правілы гульні, кулінарныя рэцэпты, метады рашэння матэматычных задач, схемы вязання і г. д.

Многія з такіх запісаў можна лічыць алгарытмамі, бо яны запісаны ў выглядзе дакладных і зразумелых каманд і прыводзяць да рашэння задач.

Існуюць наступныя спосабы запісу алгарытмаў:

- слоўнае апісанне;
- графічны (блок-схема);
- праграмны.

Слоўны спосаб запісу алгарытму — запіс алгарытму на натуральнай мове зносін.

(Разгледзьце прыклады 17.1 і 17.2, у якіх прадстаўлена слоўнае апісанне алгарытму гульні «Магія лікаў» на дзіцячым прававым сайце і алгарытму прыгатавання стравы нацыянальнай беларускай кухні).

[1] http://mir.pravo.by/welcome (дата доступу 24.01.2024).

Графічны спосаб запісу алгарытму — запіс алгарытму з дапамогай геаметрычных фігур (блокаў), якія адпавядаюць камандам алгарытму, і ліній для злучэння блокаў.

У інфарматыцы для графічнага спосабу запісу алгарытму выкарыстоўваюцца **блок-схемы,** у якіх кожны блок паказваецца ў выглядзе некаторай геаметрычнай фігуры і мае сваё прызначэнне.

Блокі пачатку і канца алгарытму: Пачатак, Канец.

Блок для запісу выконваемых каманд алгарытму: Каманда.

Блокі для ўводу зыходных даных і выводу атрыманых вынікаў: Увод, Вывад.

Запіс алгарытму ў выглядзе праграмы называецца **праграмным спосабам запісу алгарытму.**

Запісваць алгарытмы праграмным спосабам вы навучыцеся на наступных уроках.

Разгледзім наступны прыклад. Няхай у кватэры плануецца правядзенне рамонту. Мяркуецца пакрыць падлогу на кухні

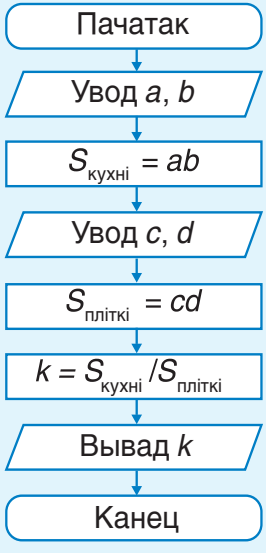
Прыклад 17.3. Рамонт на кухні.

Слоўнае апісанне алгарытму:

1. З дапамогай рулеткі вызначыць памеры кухні (даўжыню a, шырыню b).
2. Вылічыць плошчу кухні
$$S_{\text{кухні}} = ab.$$
3. З дапамогай рулеткі вымераць адну керамічную плітку (даўжыню c, шырыню d).
4. Вылічыць плошчу пліткі
$$S_{\text{пліткі}} = cd.$$
5. Вызначыць мінімальную колькасць плітак $k = \frac{S_{\text{кухні}}}{S_{\text{пліткі}}}$.

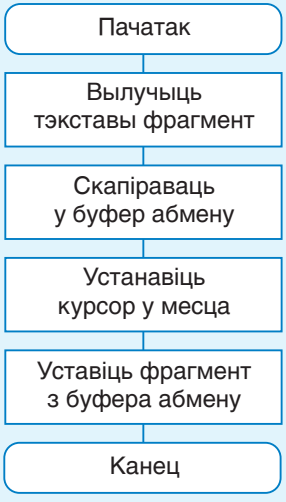
Вынікам выканання алгарытму з'яўляецца значэнне k.

Запіс алгарытму вызначэння колькасці плітак для рамонту кухні ў выглядзе блок-схемы.



Лініі са стрэлкамі на блок-схемах паказваюць на парадак выканання каманд. Калі блокі размешчаны зверху ўніз ці злева направа, то стрэлкі можна апусціць.

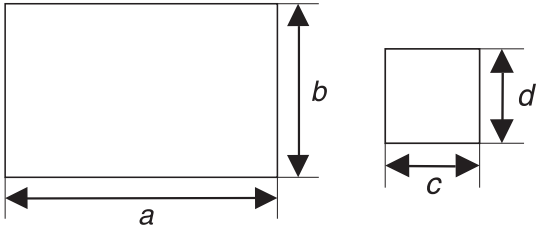
Прыклад 17.4. Графічны спосаб запісу алгарытму капіравання тэкставага фрагмента ў іншую частку дакумента.



Слоўнае апісанне каманды **Выделить текстовый фрагмент:**

1. Устанавіць тэкставы курсор у пачатак вылучаемага фрагмента.
2. Націснуць і ўтрымліваць націснутай клавішу Shift.
3. Курсорнымі клавішамі рухацца па тэксце. Націсканнем клавішы ↑ або ↓ вылучыць цэлы радок.
4. Адпусціць усе клавішы.

керамічнай пліткай. Неабходна напісаць алгарытм вызначэння мінімальнай колькасці плітак, якія спатрэбяцца для рамонту.



Слоўнае апісанне і блок-схема алгарытму, які дазваляе вызначыць неабходную колькасць плітак для рамонту, прадстаўлены ў прыкладзе 17.3.

З дапамогай рулеткі вызначым памеры кухні і пліткі і выканаем алгарытм.

1. Памеры кухні: $a = 4{,}4$ м, $b = 3{,}2$ м.

2. $S_{\text{кухні}} = 4{,}4 \cdot 3{,}2 = 14{,}08$ (м^2).

3. Памеры пліткі: $c = 0{,}33$ м, $d = 0{,}33$ м.

4. $S_{\text{пліткі}} = 0{,}33 \cdot 0{,}33 = 0{,}1089$ (м^2).

5. $k = 14{,}08/0{,}1089 \approx 129{,}29$ (пл.).

Адказ: мінімальная колькасць плітак для рамонту кухні — 130.

У прыкладзе 17.04 паказана блок-схема алгарытму капіравання тэкставага фрагмента з адной часткі дакумента ў іншую.

У алгарытме мяркуецца, што выканаўца разумее і ўмее правільна выконваць усе каманды. У адваротным выпадку неабходна больш падрабязнае апісанне асобных каманд (напрыклад, як скапіраваць вылучаны тэкст у буфер абмену).

У прыкладзе 17.5 прыведзена слоўнае апісанне алгарытму вызначэння асаблівасцей перыяду каменнага веку.

Прыклад 17.5. Алгарытм вызначэння асаблівасцей перыяду каменнага веку.

Слоўнае апісанне:

1. Выбраць перыяд каменнага веку.
2. Указаць час яго існавання.
3. Вызначыць асноўныя прылады працы.
4. Указаць спосабы здабычы харчу чалавекам.
5. Указаць найбольш важныя падзеі і з'явы перыяду.

1. Якія спосабы запісу алгарытмаў вам вядомы?
2. Які спосаб запісу называюць слоўным?
3. Што такое графічны запіс алгарытму?
4. Пералічыце блокі (геаметрычныя фігуры), якія выкарыстоўваюцца для запісу алгарытмаў.
5. Які спосаб запісу алгарытму называюць праграмным?

Практыкаванні

1 Запішыце алгарытм перамяшчэння тэкставага фрагмента з адной часткі дакумента ў іншую.

2 Прывядзіце графічны запіс алгарытмаў рашэння прыкладаў 17.1 і 17.2.

3 Складзіце алгарытм для выканання сінтаксічнага разбору простага сказу.

4 У курсе гісторыі вы азнаёміліся з этапамі самастойнай работы з гістарычнымі крыніцамі. Запішыце іх у выглядзе алгарытму.

5 Запішыце для выканаўцы *Шасцікласнік* алгарытм складання дробаў $\frac{a}{b}$ і $\frac{c}{d}$. Выканайце алгарытм для дробаў $\frac{13}{27}$ і $\frac{12}{27}$.

6 Участак зямлі прамавугольнай формы мае даўжыню a м і шырыню b м. Запішыце алгарытм вызначэння плошчы ўчастка і даўжыні плота, які спатрэбіцца для абгароджвання ўчастка. Выканайце алгарытм на прыкладзе некаторага зямельнага ўчастка.

7 У курсе біялогіі вы пазнаёміліся з будовай раслінnай клеткі. Запішыце алгарытм, які дазваляе вызначыць будову клетак скуркі цыбулі.

8 Складзіце алгарытм дзеянняў пешахода, калі чырвоны сігнал святлафора застаў яго на праезнай частцы.

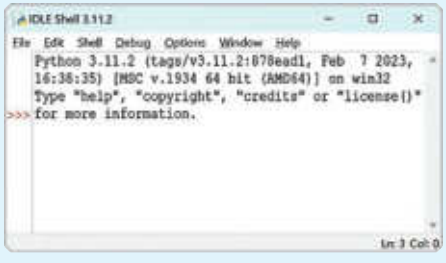
9* Запішыце алгарытм, які дазваляе вызначыць таўшчыню аркуша паперы вучэбнага дапаможніка «Інфарматыка, 6».

10* Запішыце алгарытм рашэння старадаўняй задачы: «Патрабуецца пераправіць на іншы бераг трох рыцараў і іх збраяносцаў. Ёсць лодка, якая можа змясціць толькі два чалавекі. Вядома, што ніводзін збраяносец не можа знаходзіцца побач з іншым рыцарам без яго збраяносца».

11* Ёсць збан аб'ёмам 8 л, запоўнены квасам, і два пустыя збаны аб'ёмамі 3 л і 5 л. Запішыце алгарытм, выканаўшы які, можна падзяліць квас пароўну паміж двума людзьмі (дазваляецца карыстацца толькі дадзенымі трыма збанамі).

§ 18. Асяроддзе праграмірования і камп'ютарны выканаўца

Прыклад 18.1. Асяроддзе праграмірования IDLE.



Значок асяроддзя праграмірования IDLE.

18.1. Асяроддзе праграмірования

Для распрацоўкі праграм выкарыстоўваюцца асяроддзі праграмірования (інтэграваныя асяроддзі распрацоўкі, IAP, англ. IDE, Integrated Development Environment).

Асяроддзе праграмірования — комплекс праграм, якія выкарыстоўваюцца пры распрацоўцы праграмнага забеспячэння.

Асяроддзе праграмірования можа быць разлічана на работу з ад-

ной ці некалькімі мовамі праграмірвання. Для работы з якой-небудзь мовай праграмірвання могуць выкарыстоўвацца розныя асяроддзі праграмірвання.

Для работы з мовай праграмірвання Python можна выкарыстоўваць наступныя асяроддзі праграмірвання:

- асяроддзе IDLE, якое ўстанаўліваецца разам з мовай Python[1] (прыклад 18.1);
- асяроддзе PyCharm[2] (прыклад 18.2);
- анлайн-асяроддзі[3] праграмірвання (прыклад 18.3).

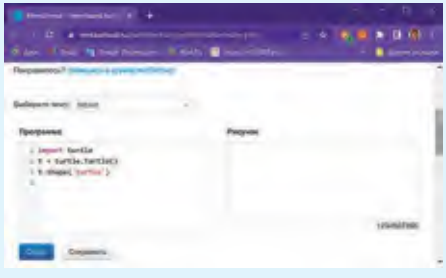
Для вывучэння асноў работы з праграмамі на мове праграмірвання Python будзем выкарыстоўваць асяроддзе праграмірвання IDLE. Запусціць яго можна з дапамогай значка або выкарыстаўшы пошукавы радок кнопкі **Пуск** (прыклад 18.4).

У асяроддзе праграмірвання ўваходзяць: рэдактар тэкстаў, даведачная сістэма, выканаўца *Чарапаха* і інш. Для стварэння сваёй праграмы трэба выканаць каманду **File** → **New File**. Будзе

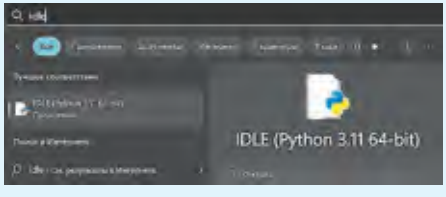
Мова **Python** з'явілася ў 1991 г. Яе распрацоўшчыкам з'яўляецца Гвіда ван Росум, галандскі праграміст.

Прыклад 18.2. Асяроддзе праграмірвання PyCharm.

Прыклад 18.3. Анлайн-асяроддзе праграмірвання выканаўцы *Чарапаха*.



Прыклад 18.4. Пошук асяроддзя праграмірвання IDLE.

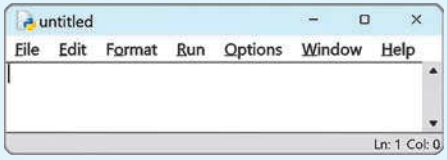


[1] https://www.python.org/downloads/

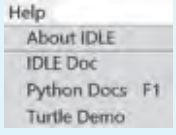
[2] https://www.jetbrains.com/ru-ru/pycharm/download/other.html. Выбраць з раздзела PyCharm Community Edition.

[3] Напрыклад, https://metaschool.ru/pub/konkurs/python/turtle/index.php

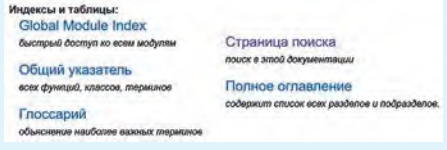
Прыклад 18.5. Акно для запісу кода праграмы.



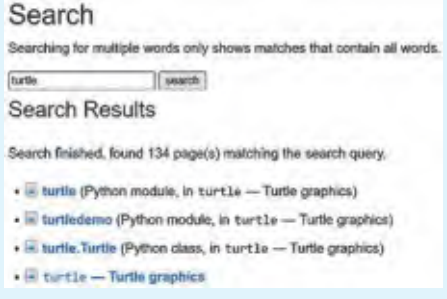
Прыклад 18.6. Меню **Help**.



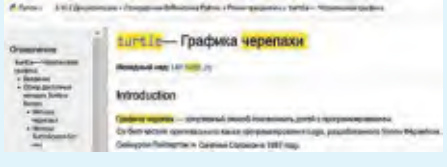
Прыклад 18.7. Выбар старонкі пошуку.



Прыклад 18.8. Пошук «turtle».



Прыклад 18.9. Частка старонкі **Графика** *Чарапахі* з перакладам на рускую мову.



створана новае акно, у якім можна пісаць праграмы (прыклад 18.05). У гэтым акне можна набіраць і рэдагаваць тэкст праграмы. Захоўваць і адкрываць файлы з тэкстамі праграм можна гэтак жа, як у тэкставым і графічным рэдактарах. Пры неабходнасці можна звярнуцца да даведачнай сістэмы праз меню **Help** (прыклад 18.6).

Дакументацыю па рабоце ў асяроддзі IDLE можна атрымаць, выканаўшы каманду **IDLE Doc**.

Дакументацыя па мове Python (каманда **Python Docs F1**) адкрываецца ў браўзеры. Дакументацыя прапануецца на англійскай мове. Перакласці на рускую мову яе можна, выкарыстаўшы кантэкставае меню (каманда **Перевести на русский**).

На старонцы **Search page/ Страница поиска** (прыклад 18.7) можна знайсці цікавую для вас інфармацыю. У прыкладзе 18.8 паказаны спіс старонак, на якіх можна атрымаць інфармацыю аб выканаўцы *Чарапаха*. Усяго знойдзена 134 старонкі. Пераход па першай спасылцы дазволіць падрабязна азнаёміцца з *Чарапахай* (прыклад 18.9), даведацца пра каманды, даступныя

выканаўцу, разгледзець прыклады праграм і іх вынікі. Прыклады можна скапіраваць у асяроддзе праграмірання і там жа выканаць. Пераклад на рускую мову даступны ў браўзеры з дапамогай кантэкставага меню.

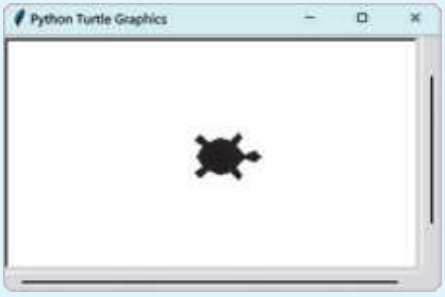
Каманда **Turtle Demo** з меню **Help** дазваляе адкрыць вялікую колькасць прыкладаў выканаўцы *Чарапаха*.

18.2. Камп'ютарны выканаўца *Чарапаха*

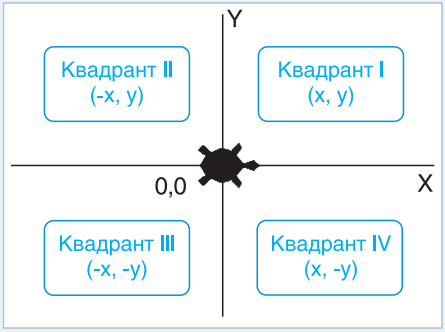
Стандартная бібліятэка Python змяшчае модуль turtle, прызначаны для навучання праграмаванню. Гэты модуль змяшчае набор функцый, якія дазваляюць кіраваць *Чарапахай*. *Чарапаха* мае пяро, якое можна паднімаць, апускаць, перамяшчаць. Пры перамяшчэнні апушчанага пяра за ёй застаецца след.

Асяроддзе існавання выканаўцы *Чарапаха* — каардынатная плоскасць (прыклад 18.10). Зыходнае месцазнаходжанне *Чарапахі* — пункт з каардынатамі (0,0), пяро апушчана. Каардынатная плоскасць *Чарапахі* па змоўчанні вызначаецца значэннямі $x = 400$, $y = 300$. Гэта значыць, што значэнні каардынаты x змяняюцца

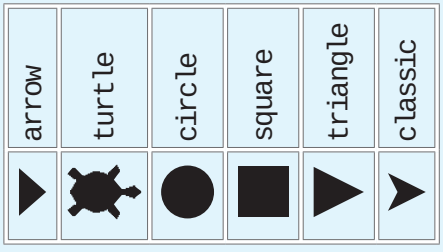
Прыклад 18.10. Асяроддзе існавання выканаўцы *Чарапаха*.



Каардынатная плоскасць *Чарапахі*, зададзеная значэннямі x і y.



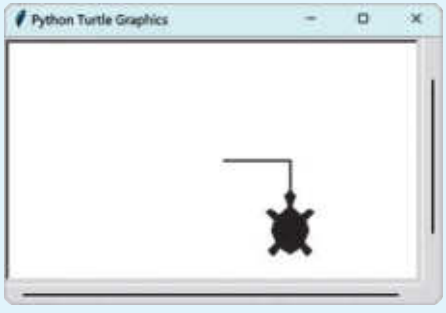
Знешні выгляд выканаўцы вызначаецца значэннем параметра `X` каманды `shape(X)`.



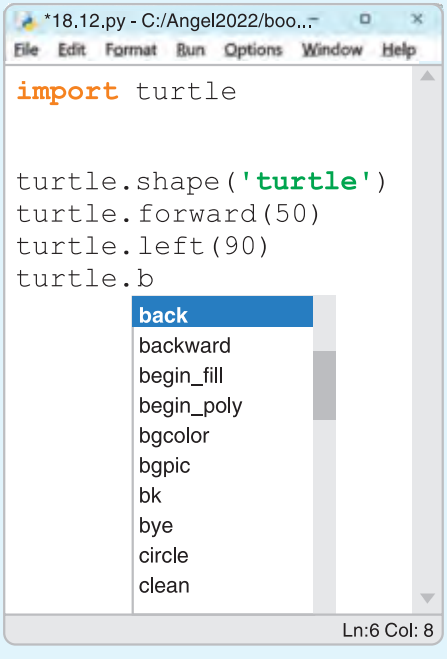
Выгляд выканаўцы можна змяніць пры запуску праграмы. Часцей за ўсё выбіраюць выгляд чарапашкі.

Прыклад 18.11. Выкарыстанне каманд `shape`, `forward`, `left`, `backward`.

```
turtle.shape('turtle')
turtle.forward(50)
turtle.left(90)
turtle.backward(40)
```



Прыклад 18.12. Выпадаючы спісак каманд.



ад −400 да 400, а каардынаты y — ад −300 да 300. Памер акна *Чарапахі* пры гэтых значэннях 800 × 600.

Некаторыя каманды выканаўцы *Чарапаха*, з дапамогай якіх можна пісаць праграмы, прадстаўлены ў табліцы.

Каманда	Дзеянне
`shape(X)`	Змяніць значок чарапахі
`penup()`	Не пакідаць след пры руху
`pendown()`	Пакідаць след пры руху
`forward(X)`	Прайсці наперад X пікселяў
`backward(X)`	Прайсці назад X пікселяў
`left(X)`	Павярнуцца налева на X градусаў
`right(X)`	Павярнуцца направа на X градусаў

(Разгледзьце прыклад 18.11.)

Для таго каб *Чарапаха* магла выконваць каманды, першай камандай у праграме трэба падключыць модуль, які змяшчае каманды кіравання *Чарапахай*:

```
import turtle
```

кожная каманда кіравання *Чарапахай* запісваецца пасля ключавога слова `turtle` і аддзяляецца

ад яго кропкай. Каманды, якія запісваюцца пасля кропкі, можна выбіраць з выпадаючага спіска.

Для адлюстравання спіска выкарыстоўваецца камбінацыя клавіш CTRL + прабел (прыклад 18.12).

Каманды *Чарапахі* можна капіраваць, выразаць і ўстаўляць, гэтак жа як у тэкставым рэдактары. Пры выкарыстанні спалучэння клавіш: Ctrl + C, Ctrl + V, Ctrl + X, Ctrl + Z — павінна быць уключана англійская мова.

Пры выкананні праграмы могуць узнікаць памылкі. *Чарапаха* выконвае каманды да першай памылкі, пасля чаго спыняецца. Пры гэтым у акне **IDLE Shell** выводзіцца адпаведнае паведамленне (прыклад 18.13): нумар радка, у якім дапушчана памылка, памылковая каманда і прапанова, як яе выправіць.

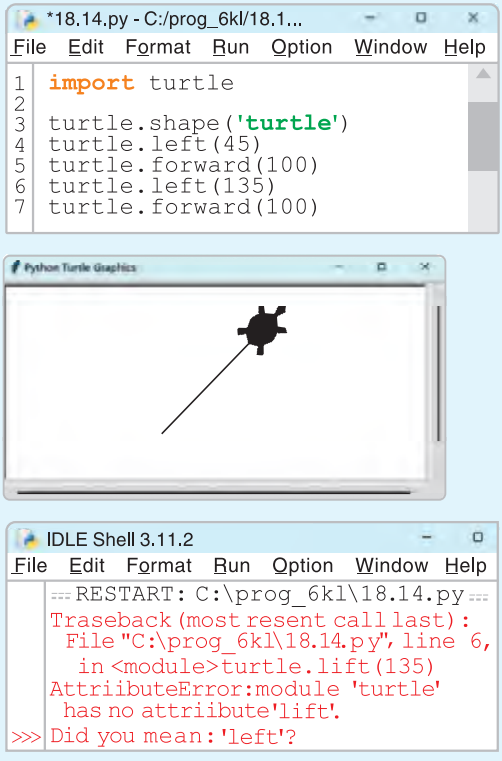
Паказаць/схаваць нумарацыю радкоў у праграме можна з дапамогай каманды **Options → Show (Hide) Line Numbers.**

Для рашэння задачы з дапамогай выканаўцы *Чарапаха* неабходна:

1. Скласці алгарытм рашэння задачы (прыклад 18.14).

2. Запісаць алгарытм у выглядзе праграмы ў акне тэкставага

Прыклад 18.13. Праграма з памылкай у радку 6 (`lift` замест `left`), графічнае акно *Чарапахі*, у якім выканана частка праграмы да з'яўлення памылкі, і рэакцыя асяроддзя IDLE на няправільную каманду.



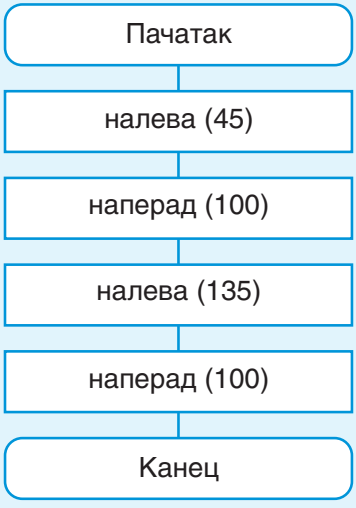
Прыклад 18.14. Алгарытм пабудовы вугла.

Слоўнае апісанне алгарытму

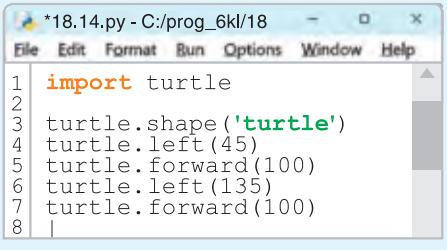
Павярнуць *Чарапаху* на 45° налева.
Зрушыць *Чарапаху* на 100 наперад.
Павярнуць *Чарапаху* на 135° налева.
Зрушыць *Чарапаху* на 100 наперад.

Прыклад 18.14. *Працяг.*

Графічны спосаб запісу алгарытму



Праграмны спосаб запісу алгарытму



Праграма заўсёды запісваецца на фармальнай мове. У слоўным і графічным апісаннях алгарытму дапускаецца натуральная мова.

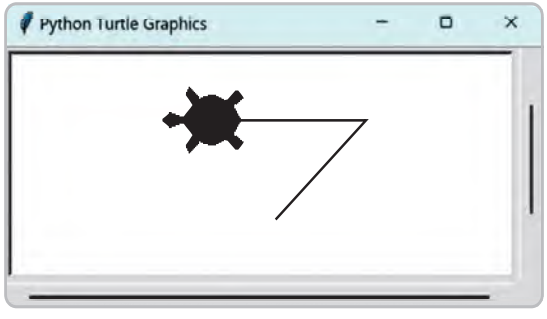
рэдактара асяроддзя праграмірання IDLE.

3. Захаваць праграму.

4. Выканаць каманду: **Run → Run Module** (можна націснуць клавішу F5).

5. Калі атрымана няправільнае рашэнне, у праграму трэба ўнесці змяненні і зноў яе выканаць.

Прыклад 18.14. Пабудаваць відарыс вугла, як на малюнку. Даўжыня адрэзкаў — 100, вугал паміж імі — 45°.



Праграма складаецца з асобных каманд. У кожным радку запісваюць адну каманду. Калі праграма не захавана перад выкананнем, то асяроддзе прапануе яе захаваць. Выконвацца можа толькі захаваная праграма.

1. Што такое асяроддзе праграмірання?
2. У якім асяроддзі праграмірання існуе камп'ютарны выканаўца *Чарапаха*?
3. Для чаго прызначаны выканаўца *Чарапаха*?

4. Якія каманды ўваходзяць у сістэму каманд выканаўцы *Чарапаха*?

5. Якое прызначэнне каманд `forward`, `backward`, `left`, `right`, `pendown`, `penup`?

Практыкаванні

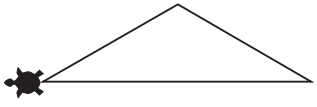
1 З дапамогай даведачнай сістэмы асяроддзя праграміравання IDLE атрымайце даведку аб камандзе мовы Python `turtle.shape`. Перакладзіце яе на беларускую мову.

2 Запішыце ў акне рэдактара асяроддзя праграміравання IDLE тэкст ніжэйпрыведзенай праграмы і вызначце вынік яе выканання.

```
import turtle
turtle.shape('turtle')
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(50)
```

3 Унясіце змяненні ў праграму з прыкладу 18.14 так, каб *Чарапаха* намалявала вугал у 60° паміж двума адрэзкамі з даўжынямі 90 і 150.

4 Складзіце праграму для атрымання відарыса трохвугольніка, праверце правільнасць выканання практыкавання.

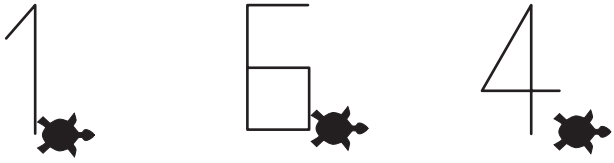


5* Напішыце каманды для пабудовы відарыса адвольнага:

а) квадрата; б) прамавугольніка; в) прамавугольнага трохвугольніка; г) раўнабедранага трохвугольніка.

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.

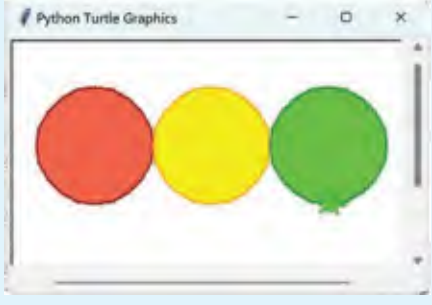
Акрамя каманд, якія былі разгледжаны ў папярэднім параграфе, у выканаўцы *Чарапаха* ёсць і іншыя. У табліцы прыведзены магчымыя каманды выканаўцы *Чарапаха*.

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.

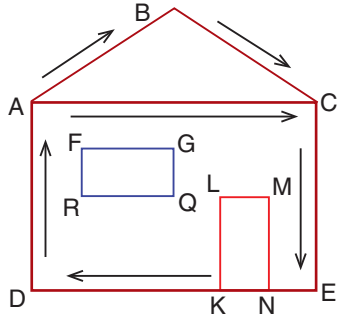
Калі тэкст праграмы вялікі, то яе складана чытаць, таму ў праграме часта пішуць каментарыі — радкі тэксту, што тлумачаць напісаныя каманды. Перад тэкстам каментарыяў ставіцца знак # (прыклад 19.3). Каментарыі можна пісаць па-беларуску. Выканаўца прапускае іх пры выкананні праграмы.

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.

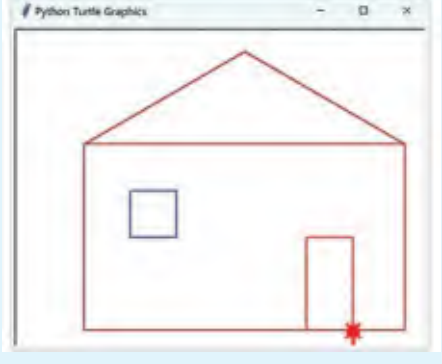
```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
turtle.penup()
turtle.setpos(-150,50)
turtle.pendown()
turtle.color('brown')
#дах
turtle.left(30)
turtle.forward(200)
turtle.right(60)
turtle.forward(200)
turtle.right(150)
turtle.forward(346)
#дом
turtle.left(90)
turtle.forward(200)
turtle.left(90)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

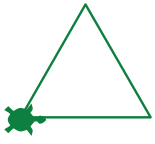
з прыкладу 19.4. Унясём змяненні ў тэкст праграмы ў акне рэдактара асяроддзя праграмірання, выкарыстаўшы правілы рэдагавання тэксту:

- 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.

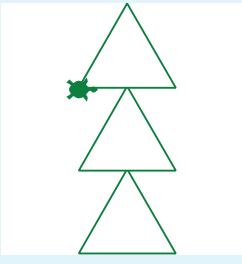
1. У гэтага трохвугольніка вуглы неаднолькавыя. *Чарапаха* павінна двойчы павярнуцца на вугал 135° (180° – 45°) і затым на вугал 90°.

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)
```

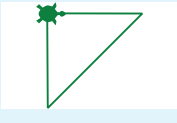
Вынік работы праграмы



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».

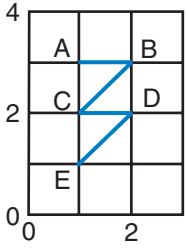
```
import turtle
turtle.shape('turtle')
turtle.pensize(2)
turtle.penup()
turtle.setpos(-50,60)
turtle.pendown()
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
turtle.forward(30)
turtle.right(135)
turtle.forward(42)
turtle.left(135)
```

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.

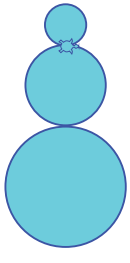
1. Як атрымаць даведачную інфармацыю аб камандах `setheading`, `pensize`?
2. Як задаць колер лініі *Чарапахі*?
3. Як зафарбаваць намаляваную *Чарапахай* фігуру?
4. Якое значэнне павінна быць у каманды `forward`, каб *Чарапаха* перамясцілася па дыяганалі квадрата з даўжынёй стараны 120?

Практыкаванні

1 Змяніце праграму з прыкладу 19.1 для атрымання відарысаў.

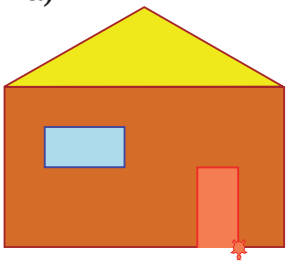
а)

б)

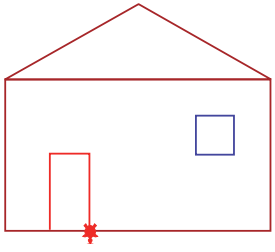


2 Змяніце праграму з прыкладу 19.4 для атрымання відарысаў. Канечнае месцазнаходжанне *Чарапахі* можа быць іншым.

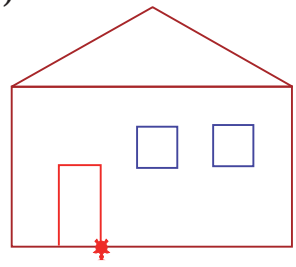
а)



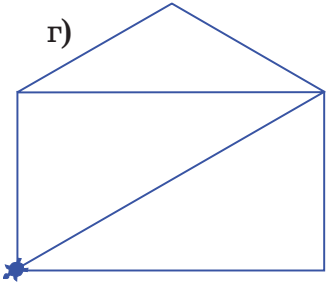
б)



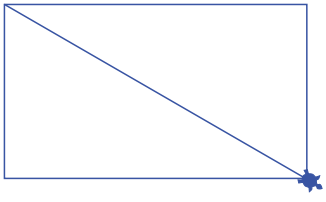
в)



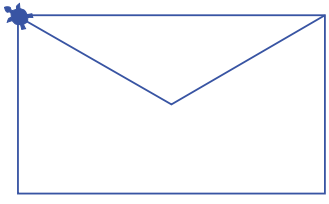
г)



д)

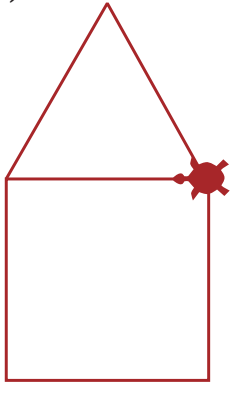


е)

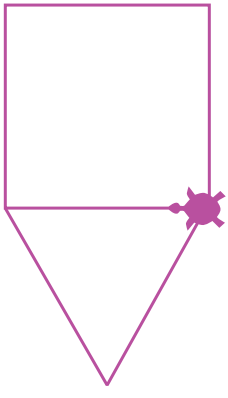


3 Выкарыстайце праграмы з прыкладаў 19.6 і 19.8 для атрымання відарысаў.

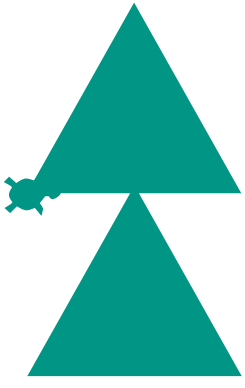
а)



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) 

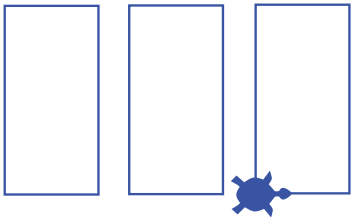
б) 

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.

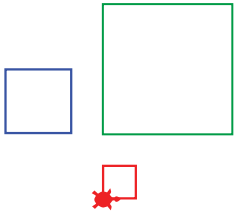
```
import turtle
turtle.shape('turtle')
turtle.color('blue')
turtle.pensize(2)
turtle.forward(60)
turtle.left(90)
turtle.forward(120)
turtle.left(90)
turtle.forward(60)
turtle.left(90)
turtle.forward(120)
turtle.left(90)
```

Змяніце праграму для пабудовы відарыса.

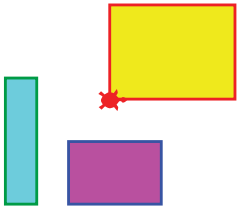


6* Складзіце праграмы для атрымання відарысаў а)—г). Падумайце, якія з ужо выкананых практыкаванняў можна выкарыстаць для атрымання гэтых відарысаў.

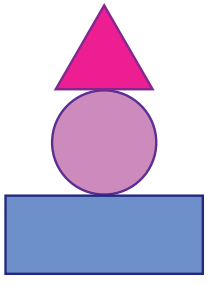
а)



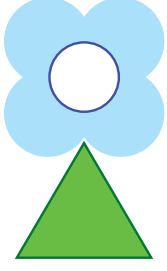
б)



в)



г)



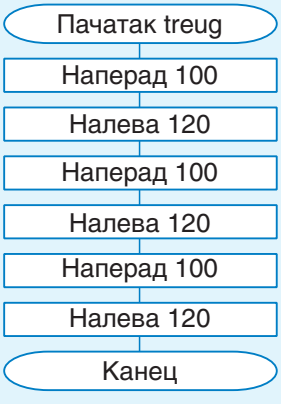
§ 20. Складанне праграм. Выкарыстанне падпраграм (дапаможных алгарытмаў)

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).

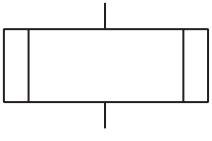


20.1. Паняцце дапаможнага алгарытму

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.

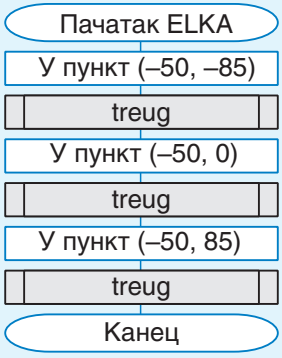
Каманду выканання дапаможнага алгарытму называюць **выклікам функцыі**. Каманду выкліку запісваюць у асноўным алгарытме (праграме) шляхам указання імя працэдуры.

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).

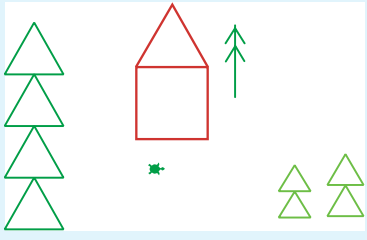
Разгледзім наступны прыклад. Няхай некалькі шасцікласнікаў атрымалі заданне распрацаваць праграму малявання некаторага «пейзажа». «Пейзаж» складаецца з наступных элементаў: дом, тры елкі (адна вялікая і дзве маленькія) і сасна (прыклад 20.6).

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).

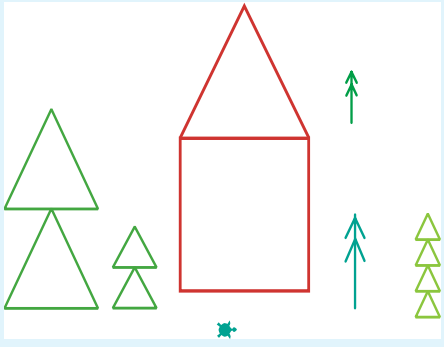
Прыклад 20.8. *Працяг.*

```
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».

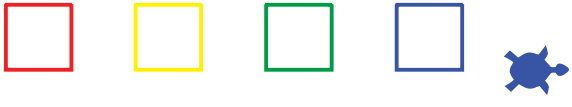
4 Складзіце праграмы пабудовы відарыса ўсіх лічбаў ад 0 да 9.

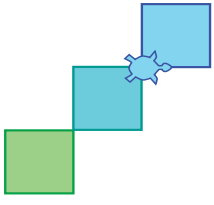
5 Запішыце праграмы напісання слоў выканаўцам *Чарапаха*. Выкарыстайце дапаможныя алгарытмы малявання літар.

а) 

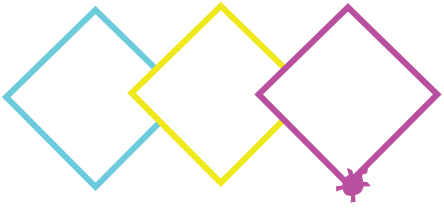
б) 

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

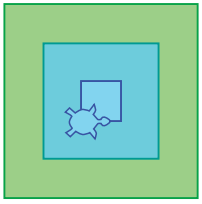
а) 

б) 

в)



г)



7 Складзіце праграму для пабудовы відарыса.