ІНФАРМАТЫКА

У. М. Котаў
А. І. Лапо
Ю. А. Быкадораў
А. М. Вайцеховіч

10

ІНФАРМАТЫКА

Вучэбны дапаможнік для 10 класа
ўстаноў агульнай сярэдняй адукацыі
з беларускай мовай навучання
(з электроннымі дадаткамі)

Дапушчана
Міністэрствам адукацыі
Рэспублікі Беларусь

Мінск «Народная асвета» 2020

УДК 004(075.3=161.3)
ББК 32.81я721
И74

Пераклад з рускай мовы *К. І. Чэрнікавай*

Аўтары:
У. М. Котаў, А. І. Лапо, Ю. А. Быкадораў, А. М. Вайцеховіч

Рэцэнзенты:
кафедра інфармацыйных тэхналогій і мадэліравання эканамічных працэсаў установы адукацыі «Беларускі дзяржаўны аграрны тэхнічны ўніверсітэт» (кандыдат педагагічных навук, дацэнт, загадчык кафедры *А. Л. Сапун*);
настаўнік інфарматыкі вышэйшай кваліфікацыйнай катэгорыі дзяржаўнай установы адукацыі «Гімназія № 1 г. Брэста» *І. Ю. Гарбацэвіч*

Электронны дадатак «Інфармацыйныя тэхналогіі» (базавы ўзровень), электронны дадатак для павышанага ўзроўню змешчаны на рэсурсе profil.adu.by

ISBN 978-985-03-3169-4

ЗМЕСТ

Ад аўтараў 6

УВОДЗІНЫ

§ 1. Алгарытм і яго ўласцівасці 8
§ 2. Мовы праграміравання 11
2.1. Высокаўзроўневыя мовы праграміравання . . . . . —
2.2. Парадыгмы праграміравання 14
2.3. Асноўныя структурныя элементы мовы праграміравання 16
Аператары 17
Даныя 18
Падпраграмы 19

Глава 1
АЛГАРЫТМЫ АПРАЦОЎКІ МАСІВАЎ

§ 3. Структураваны тып даных масіў 22
3.1. Паняцце масіву . . . . . —
3.2. Апісанне масіваў 23
3.3. Аперацыі над масівамі 24
3.4. Увод і вывад элементаў масіву 25
3.5. Рашэнне задач з выкарыстаннем уводу-выводу масіваў 27
§ 4. Выкананне арыфметычных дзеянняў над элементамі масіву 31
4.1. Вылічэнне сум і здабыткаў элементаў масіву . . . . . —
4.2. Вылічэнне сум і здабыткаў пры рабоце з двума масівамі 32
4.3. Выкарыстанне масіву, элементы якога з'яўляюцца
канстантамі 33
4.4. Пабудова кругавой дыяграмы 34

§ 5. Пошук элементаў з зададзенымі ўласцівасцямі 36
5.1. Лінейны пошук . . . . . —
5.2. Пошук аднаго элемента, які задавальняе ўмову пошуку 37
5.3. Знаходжанне ўсіх элементаў, якія задавальняюць умову пошуку 39
5.4. Рашэнне задач з выкарыстаннем алгарытму лінейнага пошуку 41
§ 6. Максімальны і мінімальны элементы масіву 48
6.1. Пошук максімальнага (мінімальнага) элемента ў масіве . . . . . —
6.2. Рашэнне задач з выкарыстаннем алгарытму пошуку максімальнага (мінімальнага) элемента 50
6.3. Пабудова гістаграмы (слупкаватай дыяграмы) 52
§ 7. Пераўтварэнне элементаў масіву 54
7.1. Асноўныя задачы . . . . . —
7.2. Змяненне элементаў масіву ў залежнасці ад выканання некаторых умоў . . . . . —
7.3. Абмен месцамі элементаў у масіве 55
7.4*. Выдаленне элемента з масіву 56
7.5*. Устаўка элемента ў масіў 57

Глава 2
КАМП’ЮТАР ЯК УНІВЕРСАЛЬНАЕ ЎСТРОЙСТВА АПРАЦОЎКІ ІНФАРМАЦЫІ

§ 8. Апаратныя сродкі камп’ютара 59
8.1. Структурная схема камп’ютара . . . . . —
8.2. Сістэмная плата, сістэмная шына, працэсар 61
8.3. Віды і прызначэнне памяці 64
§ 9. Знешнія ўстройствы 68
9.1. Класіфікацыя знешніх устройстваў . . . . . —
9.2. Апаратнае забеспячэнне для падключэння да сеткі Інтэрнэт 71
9.3. Прынцыпы работы апаратных сродкаў камп’ютара 74
§ 10. Праграмнае забеспячэнне камп’ютара 76
10.1. Праграмны прынцып работы камп’ютара . . . . . —
10.2. Розныя падыходы да класіфікацыі праграмнага забеспячэння . . . . . —

§ 11. Уяўленне даных 79
11.1. Інфармацыя і даныя . . . . . —
11.2. Аналагавае і лічбавае ўяўленне даных 81
§ 12. Кадзіраванне лікавых даных 84
12.1. Паняцце сістэмы лічэння . . . . . —
12.2. Перавод лікаў з адной пазіцыйнай сістэмы лічэння ў іншую 86
§ 13. Кадзіраванне тэкставых даных 91
13.1. Уяўленне тэксту . . . . . —
13.2. Паняцце кодавай табліцы 92
13.3. Рашэнне задач на кадзіраванне тэксту 96
§ 14. Кадзіраванне графікі, гуку і відэа 99
14.1. Кадзіраванне графікі . . . . . —
14.2. Кадзіраванне гуку 102
14.3. Кадзіраванне відэа 104
14.4. Рашэнне задач на кадзіраванне графікі, гуку і відэа 106
§ 15. Розныя падыходы да вымярэння інфармацыі 109
15.1. Змястоўны падыход . . . . . —
15.2. Алфавітны падыход 110
15.3. Імавернасны падыход 111
15.4. Рашэнне задач на вызначэнне аб’ёму інфармацыі 112
Дадатак да главы 2 114

Ад аўтараў

Дарагія старшакласнікі! Мы жывём у час хуткіх змен, калі інфарматыка становіцца найважнейшым складальнікам усёй сістэмы навуковага пазнання, вызначае шляхі фарміравання інфармацыйнага грамадства, заснаванага на ведах.

У інфарматыцы вылучаюць такія раздзелы, як тэарэтычная і прыкладная інфарматыка. У вучэбным курсе прадстаўлены абодва гэтыя раздзелы.

Тэарэтычная інфарматыка — фундаментальны складальнік інфарматыкі як вучэбнага прадмета. Агульныя заканамернасці працякання інфармацыйных працэсаў, асноўныя паняцці логікі, тэорыя алгарытмаў і мовы праграміравання — раздзелы, якія дазваляюць развіваць у навучэнцаў лагічнае і алгарытмічнае мысленне, выхоўваць інфармацыйную культуру, фарміраваць навуковы светапогляд.

Прыкладная інфарматыка накіравана на ўжыванне паняццяў і вынікаў тэарэтычнай інфарматыкі да рашэння канкрэтных задач у канкрэтных галінах. Гэты раздзел інфарматыкі ўключае ў сябе архітэктуру камп'ютара і камп'ютарных сетак, камп'ютарную графіку, камп'ютарнае мадэляванне, базы даных, інфармацыйныя тэхналогіі і інш. Вывучэнне прыкладной інфарматыкі накіравана на фарміраванне камп'ютарнай пісьменнасці — валодання неабходным наборам ведаў і навыкаў работы на камп'ютары для апрацоўкі, захоўвання, перадачы, атрымання і выкарыстання даных. Самым дынамічным складальнікам інфарматыкі з'яўляецца менавіта прыкладная інфарматыка.

Аўтары вучэбнага дапаможніка паспрабавалі зрабіць так, каб вы змаглі атрымаць неабходныя сучаснаму чалавеку веды і навыкі ва ўмовах імклівага змянення інфармацыйных тэхналогій.

Матэрыял параграфаў вучэбнага дапаможніка падзелены на дзве калонкі. Колер фону дапаможа вам разабрацца ў прызначэнні змешчанай на ім інфармацыі:

— асноўныя матэрыялы, абавязковыя для вывучэння;

— прыклады, якія ілюструюць асноўныя матэрыялы;

— азначэнні асноўных паняццяў;

— гістарычныя звесткі, інфармацыя пра вучоных, якія ўнеслі ўклад у развіццё інфарматыкі, і іншыя цікавыя факты.

У вучэбным дапаможніку выкарыстоўваюцца наступныя ўмоўныя абазначэнні:

— пытанні і заданні для праверкі ведаў;

— раздзел «Практыкаванні» змяшчае заданні, пры выкананні якіх выкарыстоўваецца камп'ютар;

— раздзел «Практыкаванні» змяшчае заданні для выканання ў сшытку;

— раздзел «Практыкаванні» змяшчае заданні, пры выкананні якіх можа быць выкарыстана інфармацыя, змешчаная на Нацыянальным адукацыйным партале;

* — практыкаванне або прыклад для цікаўных.

У тэксце некаторых практыкаванняў вам будзе прапанавана адкрыць файл. Гэта азначае, што практыкаванне можна выканаць, выкарыстоўваючы файл, змешчаны ў адпаведным курсе на сайце http://profil.adu.by (шлях: http://adu.by → «Электроннае навучанне» → сайт «Профільнае навучанне», http://profil.adu.by → катэгорыя «Информатика»/«Інфарматыка» → курс «Інфармацыйныя тэхналогіі. 10 клас (Базавы ўзровень)» або «Інфарматыка. 10 клас (Павышаны ўзровень)»). Зайсці на сайт і спампаваць файлы да практыкаванняў можна таксама з дапамогай матрычнага QR-кода.

Імя файла для спампоўвання змяшчае нумар параграфа і нумар практыкавання. Напрыклад, імя файла upr10_3 азначае, што файл належыць да трэцяга практыкавання дзясятага параграфа. Таксама на партале змешчаны файлы з праграмамі, разгледжанымі ў прыкладах. Такія файлы маюць імя pr3_1.pas (праграма для прыкладу 3.1).

У вучэбным дапаможніку змешчана шмат ілюстрацыйнага матэрыялу. Экранныя копіі прызначаны для азнаямлення з інтэрфейсамі праграм, для адлюстравання размяшчэння асобных элементаў. Падрабязна разгледзець усе структурныя элементы акна праграмы, якая вывучаецца, можна на экране камп'ютара.

Вучэбны дапаможнік для базавага ўзроўню складаецца з двух частак. У першай частцы, выдадзенай у выглядзе кнігі, размешчаны матэрыялы, якія адлюстроўваюць фундаментальны характар інфарматыкі. У электронным дадатку для базавага ўзроўню «Інфармацыйныя тэхналогіі» змяшчаюцца матэрыялы прыкладнога характару.

— дадатак «Інфармацыйныя тэхналогіі» змешчаны на электронным адукацыйным рэсурсе (http://profil.adu.by).

— матэрыял для павышанага ўзроўню змешчаны на электронным адукацыйным рэсурсе (http://profil.adu.by).

УВОДЗІНЫ

§ 1. Алгарытм і яго ўласцівасці

Тэрмін «алгарытм» паходзіць ад прозвішча матэматыка IX ст. Мухамеда ібн Муса аль-Харэзмі, які сфармуляваў правілы чатырох асноўных арыфметычных дзеянняў. Менавіта гэтыя правілы спачатку і называліся алгарытмамі, але пазней алгарытмам сталі называць любы спосаб вылічэнняў, адзіны для некаторага класа зыходных даных.

Прыклад 1.1. Алгарытм «Рэшата Эратасфена» дазваляе атрымаць простыя лікі, якія не пераўзыходзяць N.

1. Выпішам запар усе натуральныя лікі ад 2 да N.

2. Возьмем першы лік 2 і закрэслім кожны другі лік, пачынаючы адлік з наступнага за двойкай ліку.

3. Возьмем першы незакрэслены лік, большы за 2 (лік 3), і закрэслім кожны трэці лік, пачынаючы адлік ад ліку, які стаіць пасля 3 (улічваючы і раней закрэсленыя лікі).

4. Працягнем дзеянні датуль, пакуль першы незакрэслены лік не акажацца большым за $\sqrt{N}$.

5. У выніку не закрэсленымі акажуцца ўсе простыя лікі, якія не пераўзыходзяць N, і толькі яны.

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	~~49~~	~~50~~

У інфарматыку паняцце алгарытму прыйшло з матэматыкі.

Алгарытм — дакладна вызначаная сістэма зразумелых выканаўцу прадпісанняў, фармальнае выкананне якіх дазваляе атрымаць рашэнне задачы для любога дапушчальнага набору зыходных даных за канечны лік крокаў.

Прыведзенае азначэнне не з'яўляецца азначэннем у матэматычным сэнсе слова, паколькі ў ім выкарыстаны іншыя нявызначаныя паняцці — «сістэма прадпісанняў», «фармальнае выкананне» і інш. Гэта апісанне паняцця «алгарытм» раскрывае яго сутнасць. (Разгледзьце прыклад 1.1.) Апісанне можна ўдакладніць, паказаўшы агульныя ўласцівасці, якія характэрны для алгарытмаў. Да іх належаць: дыскрэтнасць, дэтэрмінаванасць (пэўнасць), зразумеласць, выніковасць, канечнасць, масавасць.

Дыскрэтнасць. Алгарытм разбіваецца на асобныя дзеянні (крокі). Выкананне чарговага дзеяння магчыма толькі пасля завяршэння папярэдняга. Пры гэтым набор прамежкавых даных канечны і атрымліваецца па пэўных правілах з даных папярэдняга дзеяння. **Каманда** з'яўляецца спецыяльным указаннем для выканаўца, якое прадпісвае выканаць кожнае асобнае дзеянне. Каманды, што залічваюць да сістэмы каманд выканаўца,

называюць простымі; іншыя каманды могуць быць вызначаны праз простыя.

Дэтэрмінаванасць. Калі алгарытм неаднаразова ўжыць у дачыненні да адных і тых жа зыходных даных, то кожны раз павінны атрымлівацца адны і тыя ж прамежкавыя вынікі і адзін і той жа выхадны вынік. Дадзеная ўласцівасць азначае, што вынік выканання алгарытму вызначаецца толькі ўваходнымі данымі і камандамі самога алгарытму і не залежыць ад выканаўца алгарытму.

Зразумеласць. Алгарытм не павінен змяшчаць каманд, сэнс якіх выканаўца можа ўспрымаць неадназначна. Запіс алгарытму павінен быць выразным, поўным і зразумелым. У выканаўцы не павінна ўзнікаць неабходнасці ў прыняцці якіх-небудзь самастойных рашэнняў.

Выніковасць. Пры дакладным выкананні каманд алгарытму вынікам павінен быць адказ на пытанне задачы. Калі спосаб атрымання наступных велічынь з якіх-небудзь зыходных не прыводзіць да выніку, то павінна быць вызначана, што трэба лічыць вынікам выканання алгарытму. У якасці аднаго з магчымых адказаў можа быць усталяванне таго факту, што задача не мае рашэння.

Канечнасць. Рэалізуемы па зададзеным алгарытме працэс павінен спыніцца праз канечны лік крокаў і выдаць шуканы вынік. Гэта ўласцівасць цесна звязана з уласцівасцю выніковасці.

Неабходнасць пабудовы фармальнага азначэння алгарытму прывяла да з'яўлення ў 20—30-х гг. XX ст. тэорыі алгарытмаў. Для вызначэння рознымі матэматыкамі былі прапанаваны:

- машына Цьюрынга;
- машына Поста;
- нармальны алгарытм Маркава.

Існавалі і іншыя азначэнні алгарытму. Пасля было даказана, што ўсе яны эквівалентныя.

Алан Мэтысан Цьюрынг (1912—1954) — англійскі логік і матэматык, які зрабіў істотны ўплыў на развіццё інфарматыкі. Прапанаваная ім у 1936 г. абстрактная вылічальная машына Цьюрынга дазволіла фармалізаваць паняцце алгарытму, якое выкарыстоўваецца ў тэарэтычных і практычных даследаваннях.

Эміль Леон Пост (1897—1954) амерыканскі матэматык і логік. Вядомы сваімі працамі па матэматычнай логіцы. Прапанаваў абстрактную вылічальную машыну — машыну Поста (1936).

Прыклад 1.2. Часта рэцэпты гатавання якіх-небудзь страў называюць алгарытмамі. У дадзеным выпадку парушаецца ўласцівасць дэтэрмінаванасці, паколькі пры гатаванні якой-небудзь стравы рознымі людзьмі вынік можа быць розным (напрыклад, ён можа залежаць ад таго, на якой пліце гатавалі, ці ад якасці прадуктаў). Акрамя таго, у рэцэптах часта бываюць фразы «пасаліць на смак», «дабавіць 2—3 лыжкі цукру» і г. д., якія парушаюць уласцівасць зразумеласці.

Прыклад 1.3. Задача можа мець рашэнне, але сфармуляваць алгарытм рашэння гэтай задачы не заўсёды атрымліваецца. Калі чалавеку прапанаваць фатаграфіі жывёл, то ён дастаткова хутка зможа падзяліць іх на дзве групы: дзікія і свойскія. Аднак сфармуляваць алгарытм, паводле якога ён гэта зрабіў, на сённяшні дзень не ўяўляецца магчымым. Некаторыя задачы класіфікацыі сёння паспяхова рашаюцца сістэмамі штучнага інтэлекту з дапамогай метадаў машыннага навучання. Аднак характэрнай рысай такіх метадаў з'яўляецца не прамое рашэнне задачы, а працэс навучання ў ходзе аналізу мноства падобных задач.

Прыклад 1.4. Спосабы праверкі алгарытму на правільнасць работы:

- матэматычны доказ;
- выкарыстанне спецыяльна падабраных тэстаў.

Масавасць. Алгарытм прыдатны для рашэння любой задачы з некаторага класа задач, г. зн. пачатковая сістэма велічынь можа выбірацца з некаторага мноства зыходных даных, якое называецца **абсягам прымянімасці алгарытму**.

Для практычнага рашэння задач на камп'ютары найбольш істотная ўласцівасць масавасці. Як правіла, каштоўнасць праграмы для карыстальніка будзе тым вышэйшай, чым большы клас аднатыпных задач яна дазволіць рашаць.

Калі распрацаваная паслядоўнасць дзеянняў не валодае хоць бы адной з пералічаных вышэй уласцівасцей, то яна не можа лічыцца алгарытмам. Для разумення ўласцівасцей алгарытму разгледзьце прыклады 1.2 і 1.3.

Для аднаго і таго ж алгарытму могуць існаваць розныя формы запісу: тэкставае апісанне, блок-схема, машына Цьюрынга і інш. Незалежна ад формы запісу любы алгарытм можа быць прадстаўлены з выкарыстаннем базавых алгарытмічных канструкцый: **паслядоўнасць, цыкл** і **галінаванне**.

У межах тэорыі алгарытмаў адбываецца аналіз розных алгарытмаў рашэння задачы для выбару найбольш эфектыўнага (аптымальнага). Распрацоўка інструментаў для аналізу эфектыўнасці алгарытмаў — адна з задач тэорыі алгарытмаў.

Любы алгарытм трэба правяраць на правільнасць работы (прыклад 1.4).

? **1.** Што такое алгарытм?
2. Якія ўласцівасці характарызуюць алгарытм?
3. Якія базавыя алгарытмічныя канструкцыі выкарыстоўваюцца пры складанні алгарытмаў?

Практыкаванні

1 Пракаменціруйце асноўныя ўласцівасці алгарытму для рэшата Эратасфена.

2 Аль-Харэзмі склаў алгарытмы арыфметычных дзеянняў яшчэ ў IX ст. Складзіце алгарытмы выканання арыфметычных дзеянняў у слупок. Праверце для складзеных алгарытмаў асноўныя ўласцівасці.

3 Прывядзіце прыклады алгарытмаў, якія валодаюць названымі прыметамі.

1. Выкарыстоўваецца толькі алгарытмічная канструкцыя *паслядоўнасць*.
2. Прысутнічае алгарытмічная канструкцыя *галінаванне*.
3. Прысутнічае алгарытмічная канструкцыя *цыкл*.
4. Выкарыстоўваюцца падпраграмы.

4 Апішыце паслядоўнасць дзеянняў для рашэння задачы «Воўк, каза і капуста».

Аднойчы селяніну спатрэбілася перавезці праз раку ваўка, казу і капусту. У селяніна ёсць лодка, у якой можа змясціцца, акрамя самога селяніна, толькі адзін аб'ект — ці воўк, ці каза, ці капуста. Калі селянін пакіне без нагляду ваўка з казой, то воўк з'есць казу; калі селянін пакіне без нагляду казу з капустай, каза з'есць капусту. Як селяніну перавезці на іншы бераг і ваўка, і казу, і капусту ў цэласці і захаванасці?

Ці будзе атрыманая паслядоўнасць дзеянняў алгарытмам? Якая ўласцівасць алгарытму не выконваецца? Ці магчыма перафармуляваць задачу так, каб аналагічная паслядоўнасць дзеянняў стала алгарытмам?

5 Ёсць два пясочныя гадзіннікі на 3 мін і на 8 мін. Як з іх дапамогай адмераць 7 мін? Вызначыце сістэму каманд выканаўца, які можа рашаць гэту задачу, і складзіце для яго паслядоўнасць дзеянняў, што прыводзіць да адказу. Якія алгарытмічныя канструкцыі былі выкарыстаны пры рэалізацыі? Ці можна лічыць атрыманую паслядоўнасць дзеянняў алгарытмам? Якая ўласцівасць алгарытму не выконваецца? Ці магчыма перафармуляваць задачу так, каб аналагічная паслядоўнасць дзеянняў стала алгарытмам?

§ 2. Мовы праграмірaвання

2.1. Высокаўзроўневыя мовы праграмірaвання

Любы алгарытм разлічаны на канкрэтнага выканаўцу, які мае сваю сістэму каманд. Алгарытм для камп'ютара павінен быць запісаны з дапамогай каманд, якія камп'ютар можа выконваць.

Першай высокаўзроўневай мовай праграмірaвання, якая была рэалізавана практычна, стаў у 1949 г. кароткі код (Short Code). Аперацыі і пераменныя ў гэтай мове праграмірaвання кадзіраваліся двухсімвальнымі спалучэннямі.

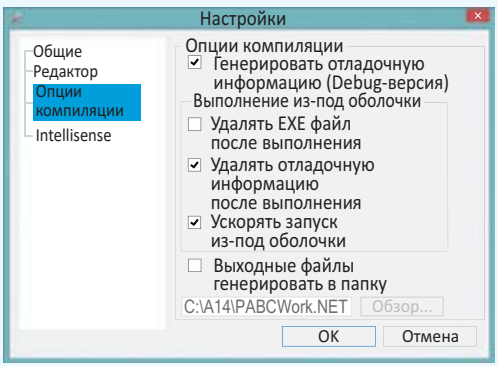
Прыклад 2.1. Некаторыя мовы праграміравання высокага ўзроўню:

- C++;
- Python;
- Pascal (Delphi);
- C#;
- Java;
- JavaScript;
- Perl;
- Fortran;
- VisualBasic;
- Lisp.

Прыклад 2.2. Пры трансляцыі праграмы адбываецца пераўтварэнне тэксту з адной мовы на іншую. Адрозніваюць наступныя віды трансляцыі: кампіляцыя і інтэрпрэтацыя.

Кампілятар — транслятар, што пераўтварае зыходны код з якой-небудзь мовы праграміравання на машынны. У выніку ствараецца файл, які можа быць выкананы непасрэдна ў аперацыйнай сістэме.

Асяроддзе праграміравання PascalABC мае ўбудаваны кампілятар, опцыі якога можна наладзіць, выканаўшы каманду **Сервис → Настройки**:



Інтэрпрэтатар — транслятар, які можа працаваць двума спосабамі:

- чытаць код і выконваць яго адразу (чыстая інтэрпрэтацыя);
- чытаць код, ствараць у памяці прамежкавае ўяўленне кода (байт-код ці р-код), выконваць прамежкавы паказ кода.

Чыстая інтэрпрэтацыя ўжываецца для моў JavaScript, VBA, Lisp і інш. Прыклады інтэрпрэтатараў, якія ствараюць байт-код: Perl, PHP, Python і інш.

Устройствам, якое выконвае каманды ў камп'ютары, з'яўляецца працэсар. Сістэму каманд працэсара называюць машыннай мовай або машынным кодам. Кожная каманда працэсара запісваецца ў двайковым кодзе. Для запісу гэтых каманд у сімвальнай форме выкарыстоўваюць мову Асэмблер. Асэмблер з'яўляецца мовай нізкага ўзроўню, паколькі змяшчае каманды, што адлюстроўваюць машынны код.

У мовах праграміравання высокага ўзроўню выкарыстоўваюцца каманды, якія аб'ядноўваюць паслядоўнасці машынных каманд. Праграмы, напісаныя на такіх мовах, прасцейшыя для разумення чалавекам, паколькі для абазначэння каманд выкарыстоўваюць словы натуральнай мовы (часцей за ўсё англійскай).

Мовы праграміравання высокага ўзроўню, або высокаўзроўневыя мовы праграміравання (прыклад 2.1), былі распрацаваны для таго, каб сутнасць алгарытму магла не залежаць ад апаратнай рэалізацыі камп'ютара. Для пераўтварэння тэксту праграмы, напісанай на мове высокага ўзроўню, у элементарныя машынныя каманды выкарыстоўваюцца спецыяльныя праграмы — **транслятары** (прыклад 2.2). Напрыклад, у тэксце праграмы дастаткова напісаць «while», а ўжо транслятар мовы перавядзе гэту каманду ў паслядоўнасць машынных кодаў. Прынцыпы работы транслятараў залежаць ад апаратнага і праграмнага забеспячэння камп'ютара.

Мовы праграміравання высокага ўзроўню з'яўляюцца фармальнымі

штучнымі мовамі. У кожнай мовы праграмiравання можна вылучыць два складальнікі: сінтаксіс і семантыку. **Сінтаксіс** (граматыка мовы) — сукупнасць правіл для напісання праграмы. **Семантыка** — сэнсавы бок мовы (вызначае сэнсавы змест моўнай канструкцыі).

Тэкст праграмы на мове высокага ўзроўню ўяўляе сабой звычайны тэкставы файл. Для яго «чытання» і ператварэння ў паслядоўнасць машынных каманд выконваецца аналіз тэксту праграмы — праверка на адпаведнасць сінтаксічным правілам і семантыцы дадзенай мовы праграмiравання. У выпадку выяўлення неадпаведнасці кампілятар можа выдаваць паведамленні пра памылкі (прыклад 2.3).

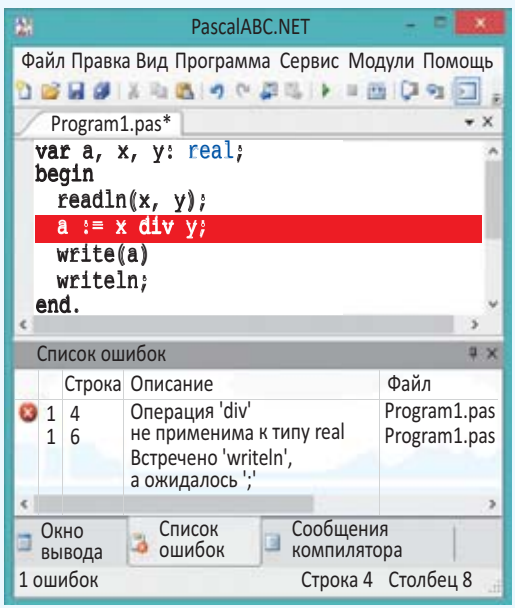
За час існавання вылічальных машын было створана больш за 8 тыс. моў праграмiравання, і штогод з'яўляюцца новыя. Некаторыя з іх належаць да вузкаспецыяльных, іншыя выкарыстоўваюцца мільёнамі праграмістаў па ўсім свеце.

Існуюць розныя падыходы да класіфікацыі моў праграмiравання. Паводле ступені адрознення семантыкі мовы ад машыннага кода мовы дзеляць на нізкаўзроўневыя і высокаўзроўневыя. Часта мовы праграмiравання падзяляюць на кампіляваныя і інтэрпрэтаваныя, аднак такое дзяленне ўмоўнае, паколькі для любой традыцыйна кампіляванай мовы (такой, як Паскаль) можна напісаць інтэрпрэтатар.

У 1951 г. амерыканскі вучоны Грэйс Мюрэй Хопер (1906—1992) стварыла першы ў свеце кампілятар і ўвяла сам гэты тэрмін.

Кампілятар ажыццяўляў функцыю аб'яднання каманд, выконваў вылучэнне памяці камп'ютара і пераўтварэнне каманд высокага ўзроўню (у той час псеўдакодаў) у машынныя каманды.

Прыклад 2.3. Семантычная (радок 4) і сінтаксічная (радок 6) памылкі ў асяроддзі PascalABC.



Роберт В. Флойд (1936—2001) — амерыканскі вучоны ў галіне тэорыі вылічальных сістэм. Лаўрэат прэміі Цьюрынга. Упершыню тэрмін «парадыгма праграміравання» быў выкарыстаны Р. Флойдам у 1978 г. у час атрымання прэміі Цьюрынга: «Калі прагрэс мастацтва праграміравання ў цэлым патрабуе пастаяннага вынаходства і ўдасканалення парадыгм, то ўдасканаленне мастацтва асобнага праграміста патрабуе, каб ён пашыраў свой рэпертуар парадыгм».

Флойд адзначаў, што парадыгмы праграміравання не з'яўляюцца ўзаемавыключальнымі, яны могуць спалучацца, узбагачаючы інструментарый праграміста.

Прыклад 2.4. Асноўная ідэя структурнага праграміравання заключаецца ў тым, што праграма павінна мець простую структуру, добра чытацца і лёгка мадыфікавацца. Структураванасць кода падтрымліваецца дзякуючы падпраграмам, якія выклікаюцца з іншых падпраграм. Структурнае праграміраванне падтрымліваюць такія мовы, як Pascal, Go, C і шмат якія іншыя.

Прыклад 2.5. Асаблівасць моў працэдурнага праграміравання складаецца ў тым, што задачы разбіваюцца на крокі і рашаюцца крок за крокам. Рашэнне для кожнага асобнага кроку афармляецца ў выглядзе асобнай працэдуры. Да працэдурных моў залічваюць: C, Pascal, Lua і інш.

2.2. Парадыгмы праграміравання

У гісторыі развіцця моў праграміравання можна вылучыць розныя парадыгмы праграміравання — сукупнасць ідэй і паняццяў, якія вызначаюць стыль праграміравання. Паміж парадыгмамі і мовамі праграміравання няма цвёрдай сувязі. Парадыгма паказвае адзін з магчымых спосабаў выкарыстання сродкаў мовы праграміравання для напісання кода праграмы. Часта мова праграміравання, створаная ў межах адной парадыгмы, праз некаторы час мадэрнізуецца, пашыраецца і пачынае выкарыстоўвацца ў межах іншай парадыгмы.

Разгледзім некаторыя парадыгмы праграміравання.

Структурнае праграміраванне — парадыгма праграміравання, у аснаве якой ляжыць уяўленне праграмы ў выглядзе блокаў іерархічнай структуры. Распрацавана ў канцы 1960-х — пачатку 1970-х гг. У адпаведнасці з дадзенай парадыгмай любая праграма складаецца з трох базавых кіруючых структур: *галінаванне, цыкл* і *паслядоўнасць*; акрамя таго, выкарыстоўваюцца падпраграмы (прыклад 2.4). Распрацоўка праграмы вядзецца пакрокава, метадам «зверху ўніз» (сыходнае праграміраванне): спачатку вызначаюцца мэты рашэння задачы, а затым ідзе дэталізацыя кожнага кроку, які, стаўшы асобнай задачай, таксама можа дэталізавацца.

Працэдурнае праграміраванне — парадыгма праграміравання, пры якой каманды, што выконваюцца паслядоўна, можна сабраць у падпраграмы

з дапамогай механізмаў самой мовы (прыклад 2.5). Гэта канцэпцыя праграмiравання «знізу ўверх», або канцэпцыя ўзыходнага праграмiравання, — распрацоўка праграм пачынаецца з распрацоўкі падпраграм (працэдур, функцый), у той час як прапрацоўка агульнай схемы не скончылася.

Парадыгмы структурнага і працэдурнага праграмiравання заснаваны на падпраграмах. Розніца заключаецца ў парадку іх распрацоўкі: «зверху ўніз» ці «знізу ўверх».

Функцыянальнае праграмiраванне — парадыгма праграмiравання, у якой працэс вылічэння тлумачыцца як вылічэнне значэнняў функцый у матэматычным разуменні. Засноўваецца функцыянальнае праграмiраванне на вылічэнні вынікаў функцый ад зыходных даных і вынікаў іншых функцый і не прадугледжвае відавочнага захоўвання стану праграмы (прыклад 2.6).

Аб'ектна-арыентаванае праграмiраванне (ААП) — парадыгма праграмiравання, заснаваная на ўяўленні праграмы ў выглядзе сукупнасці аб'ектаў і адлюстраванні іх узаемадзеяння. Канцэпцыя ААП дазваляе аб'яднаць даныя і алгарытмы іх апрацоўкі ў адзіную структуру, якую называюць класам. Кожны аб'ект з'яўляецца экземплярам якога-небудзь класа (прыклад 2.7). У ААП праграма ўяўляе сабой набор аб'ектаў, якія маюць стан і паводзіны. Стан і паводзіны аб'екта могуць змяніцца ў выніку падзеі. Праграма ў цэлым — гэта аб'ект. Для

Прыклад 2.6. У аснове функцыянальных моў ляжыць лямбда-вылічэнне (λ-вылічэнне) — матэматычная тэорыя для фармалізацыі паняцця вылічальнасці. Да іх залічваюць: Haskell, Lisp, F# і інш.

Прыклад 2.7. У праграме *тэкставы рэдактар* аб'ектамі могуць быць абзац тэксту, меню, кнопкі і г. д. У класе, які апісвае кнопку, змяшчаюцца даныя пра памер кнопкі, яе знешні выгляд, алгарытмы, якія дазваляюць «націснуць» кнопку ці «навесці на яе мыш» і інш.

Канкрэтная кнопка, напрыклад Ч, з'яўляецца аб'ектам. Такая падзея, як «пстрычка мышшу па такой кнопцы», зменіць напісанне тэксту. Падзея «двайная пстрычка мышшу па абзацы тэксту» вылучыць яго і г. д.

Да аб'ектна-арыентаваных моў залічваюць:

- C++;
- Java;
- Delphi;
- Python;
- Ruby і інш.

Развіццё аб'ектна-арыентаванага праграмiравання часта звязваюць з паняццямі «падзея» (падзейна-арыентаванае праграмiраванне) і «кампанент» (кампанентнае праграмiраванне).

Асяроддзе PascalABC дае магчымасць ствараць аконныя дадаткі. Пры распрацоўцы інтэрфейсу праграмы выкарыстоўваюцца візуальныя кампаненты, а праграмны код заснаваны на падзейным праграмiраванні. Ствараць такія дадаткі вы навучыцеся ў 11-м класе.

Прыклад 2.8. Мультыпарадыгменныя мовы праграміравання часцей за ўсё падтрымліваюць працэдурную (структурную), аб'ектна-арыентаваную і функцыянальную парадыгмы: C++, Python, JavaScript, Ruby, C#.

Існуюць і іншыя класіфікацыі і спосабы параўнання розных моў праграміравання[1].

Прыклад 2.9. Вучэбная мова забяспечвае прастату, выразнасць і лёгкачытальнасць канструкцый мовы. Як вучэбныя мовы праграміравання распрацоўваліся: Basic, Pascal, Logo, Scratch.

Прыклад 2.10. Некаторыя эзатэрычныя мовы служаць для праверкі матэматычных канцэпцый (Thue, Unlambda), іншыя ствараюцца для забавы. Часта яны парадыруюць «сапраўдныя» мовы праграміравання ці з'яўляюцца абсурдным увасабленнем канцэпцый праграміравання (Smetana, Var'aq, FiM++ і інш.).

Прыклад 2.11. Псеўдакод алгарытму знаходжання сумы квадратаў першых n натуральных лікаў.

```
ввод n;
S = 0
нц для i от 1 до n
  S = S + i * i
кц
```

Службовыя (ключавыя) словы — зарэзерваваныя словы, якія маюць спецыяльныя значэнні для кампілятара. Іх нельга выкарыстоўваць як ідэнтыфікатары ў праграмах.

выканання сваіх функцый яна звяртаецца да аб'ектаў, якія ў яе ўваходзяць. Яны, у сваю чаргу, могуць звяртацца да іншых аб'ектаў, рэалізоўваць свае метады ці рэагаваць на падзеі. Аб'ектна-арыентаванае праграміраванне асабліва важна пры рэалізацыі буйных праектаў.

Большасць сучасных моў праграміравання з'яўляюцца **мультыпарадыгменнымі** — падтрымліваюць адразу некалькі парадыгм праграміравання (прыклад 2.8).

Асобна разглядаюць такія класы моў праграміравання, як **вучэбныя** (прыклад 2.9) і **эзатэрычныя** мовы праграміравання (прыклад 2.10).

Часта для запісу алгарытмаў ужываюць **псеўдакод** — мову апісання алгарытмаў, якая выкарыстоўвае ключавыя словы моў праграміравання, але прапускае дэталі, неістотныя для разумення алгарытму (напрыклад, апісанні пераменных). Галоўная мэта выкарыстання псеўдакода — забяспечыць разуменне алгарытму чалавекам, зрабіць апісанне больш успрымальным, чым зыходны код на мове праграміравання. Пры напісанні псеўдакода можа выкарыстоўвацца лексіка рускай мовы (прыклад 2.11).

2.3. Асноўныя структурныя элементы мовы праграміравання

Для запісу элементаў мовы праграміравання выкарыстоўваецца алфавіт. **Алфавіты** большасці моў праграміравання блізкія адзін аднаму па сін-

[1] https://ru.wikipedia.org/wiki/Сравнение_языков_программирования (дата доступу: 10.02.2019).

таксісе і, як правіла, выкарыстоўваюць літары лацінскага алфавіта, арабскія лічбы і агульнапрынятыя спецсімвалы (знакі прыпынку, знакі матэматычных аперацый і параўнанняў, раздзяляльнікі, службовыя словы). Большасць распаўсюджаных моў праграмiравання змяшчаюць у сваім алфавіце наступныя элементы:

- літары — {AaBbCcDd...};
- лічбы — {0 1 2 3 4 5 6 7 8 9};
- знакі арыфметычных аперацый — {× / + -...};
- знакі параўнання — {< > = ...};
- раздзяляльнікі — {. , ; : ( ) { } [ ]... };
- службовыя словы — {**if while for** і г. д.};
- каментарыі — любы набор сімвалаў і інш.

Нягледзячы на значныя адрозненні паміж мовамі, шмат якія фундаментальныя паняцці ў большасці моў праграмiравання падобныя паміж сабой (прыклад 2.12). Паводле вядомай формулы Н. Вірта «Алгарытмы + структуры даных = праграмы», разглядаючы мову праграмiравання, трэба гаварыць пра спосабы запісу каманд алгарытму з дапамогай аператараў і арганізацыі работы з данымі (прыклад 2.13).

Аператары

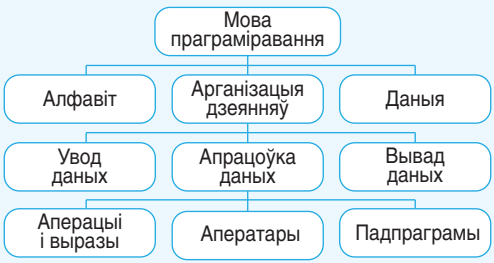
Адным з асноўных паняццяў усіх моў праграмiравання з'яўляецца аператар, які ўяўляе сабой скончаную фразу мовы і з'яўляецца прадпісаннем на выкананне пэўных дзеянняў па апрацоўцы даных. Праграма будуецца з аператараў гэтак жа, як тэкст літаратурнага твора фарміруецца са сказаў.

Прыклад 2.12. Некаторыя службовыя словы (у алфавітным парадку) у розных мовах праграмiравання.

Pascal	Python	C++

`const,`
`do,`
`else,`
`for,`
`if,`
`then,`
`var,`
`while`
	`def,`
	`elif,`
	`else,`
	`for,`
	`if,`
	`return,`
	`while`
		`const,`
		`do,`
		`else,`
		`for,`
		`if,`
		`return,`
		`while`

Прыклад 2.13. Структурная схема працэдурнай мовы праграмiравання.



Выразы будуюцца з велічынь (канстант і пераменных), функцый, дужак, знакаў аперацый і г. д. Тып выразу вызначаецца вынікам вылічэнняў. Выразы могуць прымаць лікавыя, лагічныя, сімвальныя, радковыя і іншыя значэнні.

Выраз 5 + 3 з'яўляецца лікавым, а выраз $A + B$ можа мець самы розны сэнс у залежнасці ад таго, што стаіць за ідэнтыфікатарамі A і B (калі A і B — радкі, то вынік — радок, які атрымаўся пры канкатэнацыі зыходных радкоў).

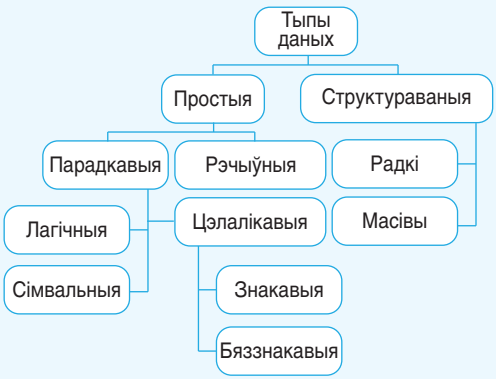
Прыклад 2.14. Запіс аператара цыкла (пошук сумы квадратаў першых *n* натуральных лікаў).

Pascal:
```
for var i := 1 to n do
  s := s + i * i;
```

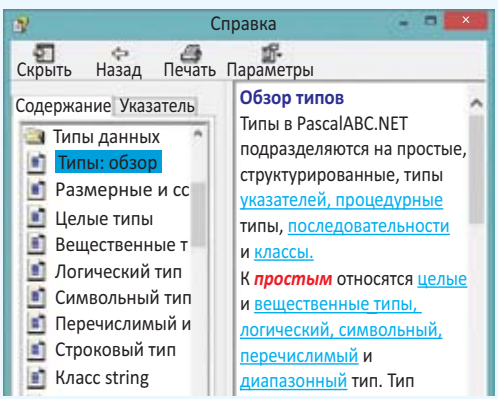
Python:
```
for i in range(n + 1):
    s += i * i
```

C++:
```
for (int i = 1; i <= n; i++)
    s += i * i;
```

Прыклад 2.15. Класіфікацыя даных у мове праграмірования.



Падобная структура тыпаў даных уласцівая шмат якім мовам праграмірования. З іншымі тыпамі даных, якія выкарыстоўваюцца ў мове PascalABC, можна пазнаёміцца ў даведачнай сістэме.



Вылучаюць наступныя аператары: аператар прысвойвання, аператар умоўнага пераходу (галінавання), аператар цыкла (прыклад 2.14), аператар выбару, састаўны аператар, часам выкарыстоўваюць пусты аператар, аператар безумоўнага пераходу і інш.

Усе аператары мовы ў тэксце праграмы аддзяляюцца адзін ад аднаго з дапамогай відавочных ці невідавочных раздзяляльнікаў (у Pascal такім раздзяляльнікам з'яўляецца «;»). Аператары выконваюцца ў тым парадку, у якім яны запісаны ў тэксце праграмы. Парадак выканання аператараў можа быць зменены толькі пры дапамозе кіруючых аператараў: галінавання, цыкла і інш.

Даныя

Большая частка аператараў прызначана для апрацоўкі велічынь. Велічыня характарызуецца тыпам, імем і значэннем. Тыпы даных могуць быць простымі і структуравanымі (прыклад 2.15). Велічыня простага тыпу ў кожны момант мае адно значэнне. Велічыня структураванага тыпу складаецца з велічынь іншых тыпаў. Напрыклад, радок складаецца з сімвалаў, кожны асобны сімвал радка мае сваё значэнне (код). Самым распаўсюджаным структураваным тыпам даных з'яўляецца масіў, з якім вы пазнаёміцеся ў гэтым вучэбным годзе.

Усім аб'ектам у мовах праграмірования (перамennым, функцыям, працэдурам і інш.) даюцца імёны. Імя аб'екта ў праграме называюць **ідэнтыфікатарам** (ад слова «ідэнтыфікаваць»).

Ідэнтыфікатарам з'яўляецца любая канечная паслядоўнасць літар і лічбаў, якая пачынаецца з літары (прыклад 2.16). Імя можа змяшчаць знак падкрэслівання «_». Выкарыстоўваць службовыя словы мовы ў якасці ідэнтыфікатара забараняецца ў большасці моў праграмiравання.

Велічыні могуць быць пастаяннымі (канстанты) і пераменнымі. **Пераменная** можа прымаць некаторае значэнне ў выніку выканання каманды ўводу ці з дапамогай аператара прысвойвання. У ходзе выканання праграмы значэнні пераменнай могуць неаднаразова змяняцца. Пасля апісання пераменная атаясамліваецца з некаторым блокам памяці, змесціва якога з'яўляецца яе значэннем. Пераменная захоўвае значэнне, якое адпавядае яе тыпу (напрыклад, пераменная цэлага тыпу не можа прымаць значэнне рэчаіснага ліку). Гэта значэнне можа здабывацца з памяці для выканання з ім аперацый, якія адпавядаюць тыпу пераменнай.

Падпраграмы

Алгарытм, які рэалізуе рашэнне асобнай часткі асноўнай задачы, называюць **дапаможным**, а яго запіс на мове праграмiравання — **падпраграмай**. Падпраграмы могуць быць рэалізаваны ў выглядзе функцый ці працэдур.

Прыклад 2.16. Імёны пераменных задае праграміст. Існуюць рэкамендацыі, як можна (трэба) называць пераменныя ў кодзе:

- імя пераменнай павінна быць зразумелым, наглядным і адлюстроўваць сутнасць абазначанага аб'екта (sum, number, count_of_positive);
- уводзіць пераменным кароткія імёны (*s*, *i*, *n*) можна ў тым выпадку, калі яны выкарыстоўваюцца ў невялікім фрагменце кода і іх ужыванне відавочна.

Некаторым ідэнтыфікатарам загадзя прадпісаны пэўны сэнс, напрыклад sin, cos, sqrt, abs — імёны матэматычных функцый.

Шмат якія кампаніі распрацоўваюць свае правілы па афармленні кода, дзе прапісаны таксама і правілы называння пераменных. У кампаніі Microsoft выкарыстоўваюць так званую «венгерскую натацыю»[1]. Вядомымі з'яўляюцца таксама:

- «вярблюджая натацыя»;
- «змяіная натацыя»;
- «шашлычная натацыя»[2].

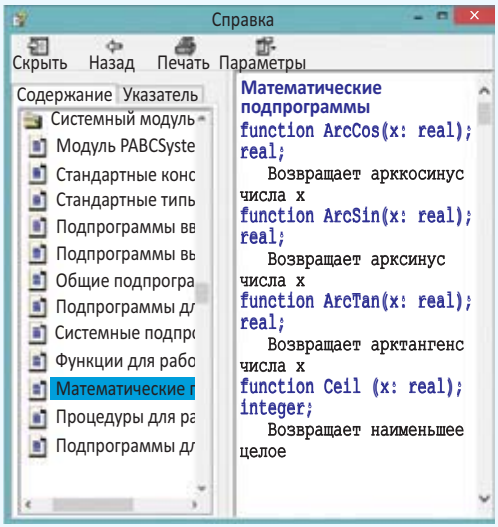
Правілы афармлення кода, распрацаваныя ў некаторых кампаніях, з'яўляюцца адкрытымі і могуць выкарыстоўвацца іншымі распрацоўшчыкамі. Напрыклад, у Google распрацаваны styleguide («гід па стылі») для розных моў праграмiравання (C++, Java, Python, Lisp і інш.)[3].

[1] https://ru.wikipedia.org/wiki/Венгерская_нотация#cite_note-hunganotat-1 (дата доступу: 25.02.2019).

[2] https://ru.hexlet.io/blog/posts/naming-in-programming (дата доступу: 25.02.2019).

[3] http://google.github.io/styleguide/ (дата доступу: 25.02.2019).

Прыклад 2.17. Спіс стандартных функцый мовы праграміравання PascalABC можна паглядзець у даведачнай сістэме:



Прыклад 2.18. Працэдуры ў мовах праграміравання.

У мове VisualBasic працэдуры аб'яўляюцца як sub (скарачэнне ад англ. *subroutine* — падпраграма).

У мове C++ няма асобных канструкцый для апісання працэдур. Іх ролю выконваюць функцыі, якія маюць тып *void* (англ. «пустата»).

На сайце Tiobe[1] штомесяц публікуецца рэйтынг моў праграміравання.

Рэйтынг складаецца на аснове падліку вынікаў пошукавых запытаў, якія змяшчаюць назву мовы, у Google, Blogger, Wikipedia, YouTube, Baidu, Yahoo!, Bing, Amazon.

Функцыя апісвае працэс вылічэння пэўнага значэння, залежнага ад некаторых аргументаў, таму для функцыі заўсёды паказваецца тып значэння, якое вяртаецца. Для кожнай мовы высокага ўзроўню распрацавана бібліятэка стандартных функцый: арыфметычных, лагічных, сімвальных і г. д. (прыклад 2.17). Функцыі (як стандартныя, так і тыя, што задаюцца праграмістам) выкарыстоўваюцца ў праграме ў выразах.

Часта выкарыстоўваюць падпраграмы, якія не вяртаюць пэўнае значэнне, а ўяўляюць сабой самастойны этап апрацоўкі даных. У мове Pascal іх называюць **працэдурамі**, у іншых мовах яны могуць называцца па-іншаму ці не мець уласнай назвы і апісвацца гэтак жа, як функцыі (прыклад 2.18).

Мова праграміравання — інструмент для рашэння пэўнай задачы. Не існуе адзінай самай лепшай мовы праграміравання. Для рашэння задач рознага роду і ўзроўню складанасці трэба ўжываць розныя мовы і тэхналогіі праграміравання. У найпрасцейшых выпадках дастаткова асвоіць асновы структурнага напісання праграм, напрыклад на мове PascalABC. Для стварэння ж складаных праектаў патрабуецца не толькі свабодна валодаць якой-небудзь мовай у поўным аб'ёме, але і мець уяўленне пра іншыя мовы і іх магчымасці. Як правіла, чым больш складаная задача, тым больш часу патрабуецца на асваенне інструментаў, неабходных для яе рашэння.

[1] https://www.tiobe.com/tiobe-index/ (дата доступу: 26.06.2019).

1. Для чаго прызначаны транслятар?
2. Якія функцыі выконвае кампілятар? Інтэрпрэтатар?
3. Што вызначаецца парадыгмай праграмiравання?
4. З якіх элементаў можа складацца алфавіт мовы праграмiравання?
5. Што ўяўляе сабой аператар мовы праграмiравання?
6. Якія тыпы даных вам вядомыя?
7. Для чаго выкарыстоўваюцца функцыі і працэдуры?

Практыкаванні

1 Напішыце праграмы для рашэння наступных задач.

1. Вызначыце апошнюю лічбу натуральнага ліку N.
2. Два адрэзкі на плоскасці задаюцца каардынатамі сваіх канцоў. Вызначыце, які з іх карацейшы.
3. Знайдзіце суму $1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots+\frac{1}{N^2}$ для зададзенага N.
4. Уводзіцца радок тэксту. Вызначыце, ці з'яўляецца ён паліндромам.
5. Уводзяцца два цэлыя лікі, якія з'яўляюцца лічнікам і назоўнікам дробу. Скараціце дробь, выведзіце атрыманыя лічнік і назоўнік. Падказка: можна выкарыстаць алгарытм Еўкліда.

6*. Дзед Мазай і заяц гуляюць у вельмі простую гульню. Перад імі — гара з N аднолькавых морквін. Кожны з гульцоў у час свайго ходу можа ўзяць з яе любую колькасць морквін, роўную неадмоўнай ступені ліку 2 (1, 2, 4, 8, ...). Гульцы ходзяць па чарзе. Хто возьме апошнюю морквіну, той і выйграе. Складзіце алгарытм, які пры зададзеным значэнні N вызначае пераможцу ў гэтай гульні. Улічвайце, што кожны з гульцоў хоча выйграць і не робіць лішніх хадоў, г. зн. гуляе аптымальна.

2* Прапанаваныя ніжэй алгарытмы запісаны рознымі спосабамі. Вызначыце, што робіць кожны з прапанаваных алгарытмаў, і рэалізуйце іх на мове Pascal.

1. Алгарытмічная мова

```
ввод a
n := Длина(a)
m := 1
b := Извлечь(a, m)
нц для i от 7 до n
  c := Извлечь(a, i)
  b := Склеить(b, c)
кц
вывод b
```

Для рускага слова «энергетика» праграма выводзіць «этика».

2. Python

```
a = int(input())
k = 0
s = 1
while k < a:
  k = k + 1
  s = s + 1.0/k
print(s)
```

Пры $a = 5$ праграма выводзіць 3.2833333333333337.

3. Basic

```
INPUT X
L = 0
M = 0
WHILE X > 0
   M = M + 1
    IF X MOD 3 <> 0 THEN
     L = L + 1
    END IF
   X = X \ 3
WEND
PRINT L
PRINT M
```

Для значэння 5637 праграма выводзіць 6 і 8.

4. C++

```
int F(int x)
{
   return x*x + 16*x + 15;
}
int main()
{
 int a, b;
 cin >> a >> b;
 int M = 0;
 for (int t = a; t <= b; t++)
  if (F(t) > 0)
    M = M + 1;
 cout << M;
 return 0;
}
```

Пры $a = -3$, $b = 5$ праграма выводзіць 6.

3* Адлюструйце любы алгарытм з практыкавання 2 у выглядзе блок-схемы.

Глава 1
АЛГАРЫТМЫ АПРАЦОЎКІ МАСІВАЎ

§ 3. Структураваны тып даных масіў

Упершыню тып даных масіў з'явіўся ў мове Фартран (створана ў перыяд з 1954 па 1957 г. у карпарацыі IBM). Ужо першыя версіі мовы падтрымлівалі трохмерныя масівы (у 1980 г. максімальная размернасць масіва была павялічана да 7). Масівы былі неабходны для стварэння матэматычных бібліятэк, у прыватнасці тых, што змяшчалі працэдуры рашэння сістэм лінейных ураўненняў.

Прыклад 3.1. У 10 А класе 25 навучэнцаў. Вядомы рост кожнага ў сантыметрах. Для захоўвання значэнняў росту можна выкарыстоўваць масіў A, які складаецца з 25 цэлых лікаў. Індэкс кожнага элемента — парадкавы нумар навучэнца са спіса ў класным журнале. Тады запіс A[5] — рост навучэнца пад пятым нумарам.

3.1. Паняцце масіву

У сучасным свеце штосекундна адбываецца апрацоўка вялізнай колькасці даных з дапамогай камп'ютара. Калі неабходна апрацоўваць даныя аднаго тыпу (лікі, сімвалы, радкі і інш.), то для іх захоўвання можна выкарыстаць тып даных, які называецца **масіў**.

Масіў — упарадкаваная паслядоўнасць даных, якая складаецца з канечнага ліку элементаў, што маюць адзін і той жа тып, і абазначаецца адным імем.

Масіў з'яўляецца структураваным (састаўным) тыпам даных. Гэта азначае, што велічыня, апісаная як масіў,

складаецца з канечнага ліку іншых велічынь. Так, напрыклад, можна стварыць масівы з 10 цэлых або 100 рэчаісных лікаў. Тып элементаў масіву называюць **базавым тыпам**. Усе элементы масіву ўпарадкаваны па індэксах, якія вызначаюць месцазнаходжанне элемента ў масіве.

Масіву прысвойваецца імя, пры запісе якога можна спасылацца на дадзены масіў як на адзінае цэлае. Элементы, што ўтвараюць масіў, упарадкаваны так, што кожнаму з іх адпавядае нумар (індэкс), які вызначае месца элемента ў агульнай паслядоўнасці (прыклады 3.1—3.3). Індэксы ўяўляюць сабой выразы любога простага тыпу, акрамя рэчыўнага. Доступ да кожнага асобнага элемента ажыццяўляецца зваротам да імені масіву з запісам індэкса патрэбнага элемента, індэкс элемента запісваецца пасля імені ў квадратных дужках (прыклад 3.4).

Калі зварот да элементаў масіву ажыццяўляецца пры дапамозе толькі аднаго індэкса, то такі масіў называюць **аднамерным** або **лінейным**. Элементы дадзенага масіву размяшчаюцца ланцужком адзін за адным. Колькасць індэксаў, па якіх звяртаюцца да элемента ў масіве, вызначае **размернасць масіву**. Акрамя аднамерных, могуць выкарыстоўвацца двухмерныя, трохмерныя і іншыя масівы.

3.2. Апісанне масіваў

Апісанне масіву ў мове Паскаль адбываецца наступным чынам:

```
var <імя масіву>: array[<тып індэкса>] of <тып элементаў>;
```

Прыклад 3.2. У 10 Б класе 27 навучэнцаў. У класным журнале запісаны прозвішча і імя кожнага з іх. Для захоўвання спіса навучэнцаў можна выкарыстоўваць масіў S, які складаецца з 27 радкоў. Індэкс кожнага элемента — парадкавы нумар навучэнца са спіса ў класным журнале. Тады запіс `S[5]` — прозвішча і імя навучэнца пад нумарам 5.

Прыклад 3.3. Кожны дзень у снежні вымяралі тэмпературу паветра. Для захоўвання значэнняў тэмпературы можна выкарыстоўваць масіў `T`, які складаецца з 31 рэчаіснага ліку. Індэкс элемента — нумар дня ў снежні. Запіс `T[15]` — тэмпература паветра 15 снежня.

Прыклад 3.4. Зварот да элемента масіва: `a[3]`, `T[i]`, `S[n-1]`.

Двухмерны масіў — масіў, элементамі якога з'яўляюцца аднамерныя масівы. Яго можна ўявіць як табліцу з данымі, у якой кожны радок — лінейны масіў. Зварот да элемента ажыццяўляецца па двух індэксах: `a[3][5]` — элемент, змешчаны ў трэцім радку і пятым слупку. Прыкладам выкарыстання двухмернага масіву з'яўляецца ліст электроннай табліцы.

Масіў p, апісаны наступным чынам:

```
var p: array [1..30] of array [1..30]
       of boolean;
```

можна выкарыстоўваць для вызначэння свабодных месцаў у глядзельнай зале. Тады запіс

```
p[2][13] := true;
```

будзе абазначаць, што ў другім радзе месца 13 свабоднае, а запіс

```
p[5][7] := false;
```

— у пятым радзе месца 7 занятае.

Прыклад 3.5. Апішам масіў, разгледжаны ў прыкладзе 3.1. Памер апісанага масіву — 25 элементаў:

```
var A: array[1..25] of integer;
```

Прыклад 3.6. Апішам масіў, разгледжаны ў прыкладзе 3.2:

```
var S: array[1..27] of string;
```

Прыклад 3.7. Апішам масіў, разгледжаны ў прыкладзе 3.3. Памер апісанага масіву — 31 элемент.

```
var T: array[1..31] of real;
```

Прыклад 3.8*. Апісаць масіў для захоўвання наступных даных: маецца будынак склада, у якім ёсць два падземныя паверхі, цокальны і тры верхнія. Неабходна захоўваць колькасць пустых адсекаў склада на кожным паверсе. Памер апісанага масіву — 6 элементаў:

```
const verh = 3;
      niz = -2;
var Sklad: array[niz..verh] of
                          integer;
```

Прыклад 3.9. Каманда `b := a;` дапушчальная для масіваў `a` і `b`, апісаных наступным чынам:

```
var a, b: array[1..10] of integer;
```

Але каманда `b := a;` выдасць памылку, калі масівы будуць апісаны так:

```
var a: array[1..10] of integer;
    b: array[1..10] of integer;
```

або так:

```
var a: array[1..10] of integer;
    b: array[1..15] of integer;
```

або так:

```
var a: array[1..10] of integer;
    b: array[1..10] of real;
```

Імя масіву з'яўляецца ідэнтыфікатарам і задаецца па тых жа правілах, што і імёны любых іншых пераменных.

Тып індэкса вызначае, як будуць нумаравацца элементы ў масіве. Для задання тыпу індэкса паказваюць нумар першага элемента ў масіве, затым ставяць дзве кропкі, пасля якіх пішуць нумар апошняга элемента. Даныя, якія выкарыстоўваюцца для задання індэксаў, павінны быць канстантамі. Дыяпазон індэксаў вызначае максімальна магчымая колькасць элементаў у масіве — **памер масіву.**

Тып элементаў задае значэнне базавага тыпу для дадзенага масіву. Базавы тып можа быць любым з вядомых вам тыпаў (прыклады 3.5—3.8).

3.3. Аперацыі над масівамі

Масівы, апісаныя аднолькава (у адной камандзе апісання), можна выкарыстоўваць у аперацыях прысвойвання. У выніку выканання гэтай каманды ўсе элементы аднаго масіву будуць перапісаны ў другі (прыклад 3.9). Калі масівы апісаны аднолькава, але ў розных радках ці апісаны па-рознаму, то пры спробе прысвойвання ўзнікне памылка пра немагчымасць пераўтварыць тыпы.

Ніякія іншыя аперацыі для масіву як для тыпу даных не вызначаны.

Аперацыі, якія выконваюцца з элементамі масіву, адпавядаюць аперацыям, выкананым над базавым тыпам. Калі, напрыклад, апісаны масіў з лікаў тыпу `integer`, то з элементамі такога масіву можна выконваць такія ж аперацыі, як і з цэлымі ліка-

мі. Элементы масіву называюць **індэксаванымі пераменнымі**. Яны могуць выкарыстоўвацца так жа, як і простыя пераменныя (прыклад 3.10).

3.4. Увод і вывад элементаў масіву

Каб працаваць з масівам, неабходна задаць пачатковыя значэнні элементаў масіву. Зрабіць гэта можна некалькімі спосабамі:

1) увод элементаў масіву з клавіятуры;

2) выкарыстанне выпадковых лікаў для вызначэння значэнняў;

3) выкарыстанне функцый (стандартных ці ўласных) для вызначэння значэнняў;

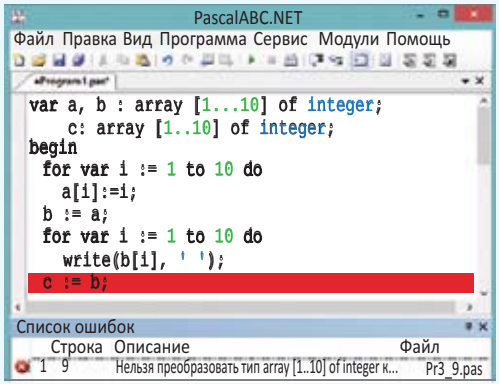
4) вызначэнне элементаў масіву як канстант.

Пры ўводзе элементаў масіву з клавіятуры кожны элемент павінен уводзіцца асобна. Калі колькасць элементаў, якія ўводзяцца, вызначана, то можна выкарыстаць цыкл **for** (прыклад 3.11).

Пры ўводзе элементаў масіву неабходна памятаць, што колькасць элементаў, якія ўводзяцца, не можа быць большай за памер масіву. У масіў, апісаны ў прыкладзе 3.11, можна ўвесці любую колькасць лікаў ад 1 да 10, змяніўшы значэнне 10 у загалоўку цыкла.

Пры апісанні масіву памер вызначае максімальную колькасць магчымых элементаў. Пры ўводзе можна вызначаць колькасць элементаў, якая неабходна для апрацоўкі ў кожным пэўным выпадку (прыклад 3.12).

Прыклад 3.9. *Працяг.*



Прыклад 3.10. Аперацыі над індэксаванымі пераменнымі:

```
a[3] := 25 mod 7;
s := (t[1] + t[30])/2;
a[k] := b[k]*2;
Sum := Sum + a[i];
if a[i] < 0 then ...
```

Прыклад 3.11. Увесці 10 элементаў масіву a.

```
var a: array[1..10] of integer;
begin
 writeln('Увядзіце 10 лікаў
         праз прабел');
 for var i := 1 to 10 do
  read(a[i]);
...
end.
```

Прыклад 3.12. Увесці зададзеную колькасць элементаў масіву a.

```
var a: array[1..100] of integer;
  n: integer;
begin
 writeln('Увядзіце  колькасць
         лікаў у масіве');
 readln(n);
 writeln('Увядзіце', n,
        'лікаў праз прабел');
 for var i := 1 to n do
  read(a[i]);
...
end.
```

Прыклад 3.13. Увод масіву радкоў.

```
var a: array[1..100] of string;
    n: integer;
begin
 writeln('Увядзіце колькасць
        радкоў у масіве');
 readln(n);
 writeln('Увязіце', n,'радкоў,
       кожны з новага радка');
 for var i := 1 to n do
  readln(a[i]);
...
end.
```

Прыклад 3.14. Выпадковым чынам задаць n элементаў масіву a. Кожны элемент — лік з адрэзка [0; 100].

```
var a: array[1..100] of integer;
    n: integer;
begin
 writeln('Увядзіце колькасць
        лікаў у масіве');
 readln(n);
 for var i := 1 to n do
  a[i] := random(101);
...
end.
```

Прыклад 3.15. Апісанне масіву, элементы якога з'яўляюцца лікавымі канстантамі.

```
const simple_numb: array[1..5]
      of integer = (2, 3, 5, 7, 11);
```

Прыклад 3.16. Апісанне масіву, элементы якога з'яўляюцца радковымі канстантамі.

```
const c_rgb: array of string =
    ('чырвоны','сіні','зялёны');
```

Прыклад 3.17. Вывад элементаў масіву ў слупок (па адным у слупку).

```
for var i := 1 to n do
  writeln(a[i]);
```

Прыклад 3.18. Вывад элементаў масіву ў радок (праз прабел).

```
for var i := 1 to n do
  write(a[i], ' ');
```

Калі неабходна ўводзіць масіў з радкоў, трэба памятаць, што кожны элемент уводзіцца ў асобным радку з выкарыстаннем каманды readln (прыклад 3.13). Выкарыстоўваць прабел як раздзяляльнік не атрымаецца, паколькі прабел будзе ўспрыняты як чарговы сімвал радка.

Часам бывае зручна задаваць элементы масіву выпадковым чынам. Для гэтага выкарыстоўваецца функцыя random(k), якая генерыруе выпадковы цэлы лік з прамежку [0; k) (прыклад 3.14).

Калі элементы масіву павінны належаць адрэзку [a; b], то можна выкарыстоўваць функцыю random(a, b) ці вызначыць значэнне элемента масіву так:

```
a[i]:= random(b-a+1)+ a;
```

Калі элементы масіву не будуць змяняцца пры рашэнні задачы, то масіў можа быць апісаны як канстанта (прыклады 3.15, 3.16). Пры такім апісанні можна не паказваць індэксы элементаў у масіве, тады нумарацыя будзе ажыццяўляцца ад нуля да колькасці элементаў у спісе мінус адзін.

Выводзіць элементы масіву можна ў слупок (прыклад 3.17) ці ў радок (прыклад 3.18). Калі элементы масіву выводзяцца ў радок, то паміж імі трэба выводзіць сімвал-раздзяляльнік (часцей за ўсё выкарыстоўваюць прабел), інакш усе лікі будуць раздрукаваны запар як адзін лік з вялікай колькасцю лічбаў. Выводзіць элементы масіву можна не толькі ў

прамым, але і ў адваротным парадку (прыклад 3.19).

3.5. Рашэнне задач з выкарыстаннем уводу-выводу масіваў

Прыклад 3.20. Напісаць праграму, якая ўвядзе элементы масіву з клавіятуры і выведзе суму трэцяга і пятага элементаў.

Этапы выканання задання

I. Зыходныя даныя: масіў a і колькасць элементаў n.

II. Вынік: S — сума трэцяга і пятага элементаў.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. Вылічэнне сумы.
3. Вывад выніку.

IV. Апісанне пераменных: a — **array**[1..10] **of** integer; n, S — integer.

Прыклад 3.21. Напісаць праграму, якая сфарміруе масіў з n лікаў з адрэзка [0; 100] выпадковым чынам. Вывесці масіў на экран.

Этапы выканання задання

I. Зыходныя даныя: масіў a і колькасць элементаў n.

II. Вынік: атрыманы масіў.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. Генерацыя масіву.
3. Вывад выніку.

IV. Апісанне пераменных: a — **array**[1..100] **of** integer; n — integer.

Прыклад 3.19. Вывад элементаў масіву ў радок (у адваротным парадку).

```
for var i := n downto 1 do
  write(a[i], ' ');
```

Прыклад 3.20.
V. Праграма:

```
var a: array[1..10] of integer;
   n, S: integer;
begin
 writeln('Увядзіце колькасць
        лікаў у масіве >=5');
 readln(n);
 writeln('Увядзіце ', n,
        'лікаў праз прабел');
 for var i := 1 to n do
  read(a[i]);
 S := a[3] + a[5];
 write('Сума лікаў = ', S);
end.
```

VI. Тэсціраванне.

Окно вывода

```
Увядзіце колькасць лікаў у масіве
7
Увядзіце 7 лікаў праз прабел
12 3 4 2 1 19 7
Сума лікаў = 5
```

VII. Аналіз вынікаў. Трэці элемент масіву роўны 4, пяты элемент роўны 1, сума элементаў роўна 5.

Прыклад 3.21.
V. Праграма:

```
var a: array[1..100] of integer;
    n: integer;
begin
writeln('Увядзіце колькасць
        лікаў у масіве');
 readln(n);
 for var i:=1 to n do
  a[i] := random(101);
 for var i := 1 to n do
  write(a[i],' ');
end.
```

VI. Тэсціраванне:

Окно вывода

```
Увядзіце колькасць лікаў у масіве
8
66 44 80 3 100 66 3 78
```

Прыклад 3.22.

V. Праграма:

```
var a: array[1..100] of integer;
      n, k: integer;
begin
 writeln('Увядзіце колькасць
        лікаў у масіве');
 readln(n);
 for var i:=1 to n do
 begin
  a[i] := 2*random(10, 35);
  write(a[i], ' ');
 end;
 writeln;
 writeln('Увядзіце k');
 readln(k);
 write(a[k]);
end.
```

VI. Тэсціраванне:

Окно вывода

```
Увядзіце колькасць лікаў у масіве
7
50 56 66 60 32 24 66
Увядзіце k
5
32
```

Прыклад 3.23.

V. Праграма:

```
var s: array [1..20] of string;
   n, k1, k2: integer;
begin
 writeln('Колькасць навучэнцаў ');
 readln(n);
 writeln('Прозвішчы');
 for var i := 1 to n do
  readln(s[i]);
 writeln('k1 i k2');
 readln(k1, k2);
 for var i := k1 to k2 do
  writeln(s[i]);
end.
```

Прыклад 3.22. Напісаць праграму, якая сфарміруе масіў з n цотных лікаў з адрэзка [20; 70] выпадковым чынам. Вывесці на экран k-ы элемент масіву.

Этапы выканання задання

I. Зыходныя даныя: масіў a і колькасць элементаў n.

II. Вынік: шуканы элемент.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. Генерацыя масіву.

1.1. Каб элементы масіву былі толькі цотнымі, неабходна кожны атрыманы элемент памнажаць на 2.

1.2. Паколькі элементы памнажаюцца на два, межы зыходнага адрэзка трэба паменшыць у два разы.

1.3. Вывад масіву па элементах.

3. Увод значэння k і вывад выніку.

IV. Апісанне пераменных: a — **array**[1..100] **of** integer; n, k — integer.

Прыклад 3.23. Напісаць праграму, якая ўвядзе з клавіятуры спіс прозвішчаў навучэнцаў і выведзе з яго прозвішчы з нумарамі ад k1 да k2.

Этапы выканання задання

I. Зыходныя даныя: масіў s і колькасць навучэнцаў n, нумары прозвішчаў — k1 і k2.

II. Вынік: спіс зададзеных прозвішчаў.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. Вывад выніку.

IV. Апісанне пераменных: s — **array**[1..20] **of** string; n, k1, k2 – integer.

Прыклад 3.24. Задаць выпадковым чынам два масівы X і Y, якія змяшчаюць па n лікаў з адрэзка [100; 300], і масіў R, які змяшчае n лікаў з адрэзка [5; 100]. Пабудаваць на экране акружнасці, каардынаты цэнтраў якіх захоўваюцца ў масівах X і Y, а радыусы ў масіве R.

Этапы выканання задання

I. Зыходныя даныя: масівы X, Y, R і колькасць элементаў n.

II. Вынік: рысунак n акружнасцей.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. Генерацыя масіваў.
3. Установка прарыстага стылю заліўкі для таго, каб адлюстроўваліся акружнасці, а не кругі.
4. Вывад выніку.

IV. Апісанне пераменных: X, Y, R — **array**[1..100] **of** integer; n — integer.

Усе разгледжаныя вышэй спосабы ўводу і выводу масіваў універсальныя і могуць выкарыстоўвацца для розных кампілятараў мовы Pascal. У асяроддзі PascalABC.Net дадаткова рэалізаваны каманды print і println, з дапамогай якіх масіў можна вывесці без выкарыстання каманды цыкла. Каманда print(b); выведзе элементы масіву b у квадратных дужках праз коску: [1,2,3,4,5,6].

Каманда println пасля выводу масіву дадаткова пераводзіць курсор на новы радок. Элементы выводзяцца гэтак жа, як і пры выкарыстанні каманды print.

Прыклад 3.23. *Працяг.*

VI. Тэсціраванне:

Окно вывода

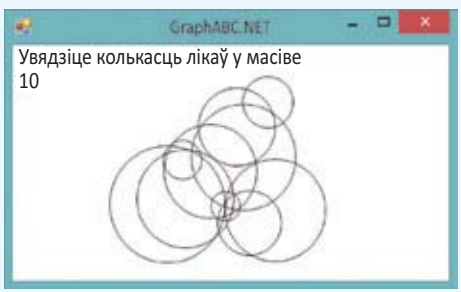
```
Колькасць навучэнцаў
6
Прозвішчы
Бялоў
Іваноў
Каралёў
Пятроў
Сідараў
Яшкін
k1 i k2
3
5
Каралёў
Пятроў
Сідараў
```

Прыклад 3.24.

V. Праграма:

```
uses graphABC;
var X, Y, R: array[1..100]
      of integer;
  n: integer;
begin
 SetWindowSize(400,400);
 writeln('Увядзіце колькасць
         лікаў у масіве');
 readln(n);
 writeln(n);
 for var i := 1 to n do
 begin
  X[i]:= random(100,300);
  Y[i]:= random(100,300);
  R[i]:= random(5,100);
 end;
 SetBrushStyle(bsClear);
 for var i := 1 to n do
   circle(X[i],Y[i],R[i])
end.
```

VI. Тэсціраванне.



1. Што такое масіў?
2. Як апісваюцца масівы?
3. Што такое памер масіву?
4. Якія аперацыі дапушчальныя для масіваў?
5. Якія спосабы задання значэнняў элементам масіву вы ведаеце?
6. Як можна вывесці масіў?

Практыкаванні

1 Выкарыстоўваючы прыклады 3.14—3.18, выканайце наступныя заданні.

1. Увядзіце 5 лікаў і выведзіце іх у адным радку.
2. Увядзіце 7 лікаў і выведзіце іх у адным радку ў адваротным парадку.
3. Задайце 10 выпадковых лікаў і выведзіце іх па адным у радку.
4. Выведзіце на экран элементы масіву, зададзенага ў прыкладзе 3.16.

2 Змяніце праграму з прыкладу 3.19 так, каб выводзіўся здабытак першых трох элементаў.

3 Выкарыстоўваючы праграмы з прыкладу 3.19 або 3.20, задайце масіў з n выпадковых лікаў з адрэзка [–10; 10]. Выведзіце: першы элемент; апошні элемент; элемент, які стаіць на сярэднім месцы.

4 Увядзіце масіў з n радкоў з клавіятуры. Выведзіце элементы масіву ў адваротным парадку.

5 Для масіву, апісанага ў прыкладзе 3.2, увядзіце даныя з клавіятуры. Задайце нумар навучэнца. Выведзіце яго прозвішча.

6* Увядзіце рост навучэнцаў свайго класа, арганізаваўшы ўвод наступным чынам:

```
Увядзіце колькасць навучэнцаў у класе: 15
Уводзьце рост навучэнцаў
навучэнец  нумар 1: 165
навучэнец  нумар 2: 170
навучэнец  нумар 3: 156
```

7* Для масіву, апісанага ў прыкладзе 3.3, задайце значэнні выпадковымі рэчыўнымі лікамі з інтэрвалу (–20; 10). Выведзіце значэнні тэмператур для зададзенага дыяпазону дат. Прыклад выводу для дыяпазону дат ад 1 снежня да 8 снежня:

```
1 снежня тэмпература была = 9.4
2 снежня тэмпература была = -11.8
3 снежня тэмпература была = -16.6
4 снежня тэмпература была = 8
5 снежня тэмпература была = 0.9
6 снежня тэмпература была = -9.3
7 снежня тэмпература была = -11.5
8 снежня тэмпература была = 6.6
```

8 Змяніце праграму з прыкладу 3.24 так, каб акружнасці маляваліся рознымі колерамі.

§ 4. Выкананне арыфметычных дзеянняў над элементамі масіву

4.1. Вылічэнне сум і здабыткаў элементаў масіву

Аперацыі, якія выконваюцца з элементамі масіву, адпавядаюць аперацыям, якія выконваюцца над базавым тыпам элементаў масіву (прыклад 4.1).

Прыклад 4.2. Зададзены аднамерны масіў з цэлых лікаў. Знайсці суму і здабытак элементаў гэтага масіву.

I. Зыходныя даныя: масіў a і колькасць элементаў n.

II. Вынік: S — сума элементаў і P — здабытак элементаў масіву.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных. Масіў уводзіцца паэлементна з клавіятуры.

2. Вызначэнне пачатковага значэння для сумы (S := 0) і для здабытку (P := 1).

3. У цыкле дабаўляем чарговы элемент масіву да сумы і да здабытку.

4. Вывад выніку.

IV. Апісанне пераменных: a — **array**[1..10] **of** integer; n, S, P — integer.

Прыклад 4.3. Вядомыя адзнакі па інфарматыцы ўсіх навучэнцаў 10 Б класа за першую чвэрць. Паспяховасць у класе будзем лічыць добрай, калі сярэдні бал большы за 7, дрэнай, калі сярэдні бал ніжэйшы за 4, у астатніх выпадках — паспяховасць сярэдняя. Вызначыць паспяховасць класа па зададзеных адзнаках.

I. Зыходныя даныя: масіў a для захоўвання адзнак і колькасць навучэнцаў n.

Прыклад 4.1.

Калі базавым тыпам элементаў масіву з'яўляецца тып integer, то для элементаў масіву дапушчальныя наступныя аперацыі: +, –, *, **div**, **mod**.

Калі ў масіве захоўваюцца лікі тыпу real, то дапушчальнымі будуць аперацыі +, –, *, /.

Калі ў масіве захоўваюцца радкі, то для кожнага яго элемента дапушчальныя радковыя функцыі і працэдуры.

Прыклад 4.2.

V. Праграма:

```
var a: array[1..10] of integer;
   n, S, P: integer;
begin
 write('Увядзіце n =');
 readln(n);
 writeln('Увядзіце элементы');
 for var i := 1 to n do
  read(a[i]);
 S := 0;
 P := 1;
 for var i := 1 to n do
 begin
  S := S + a[i];
  P := P * a[i];
 end;
 writeln('Сума = ', S);
 writeln('Здабытак = ', P);
end.
```

VI. Тэсціраванне.

```
Окно вывода
Увядзіце n = 5
Увядзіце элементы
3 2 44 -1 3
Сума = 51
Здабытак = -792
```

VII. Аналіз вынікаў. Праверыць правільнасць вылічэнняў можна на калькулятары.

Прыклад 4.3.
V. Праграма:

```
var a: array[1..30] of integer;
 n, S: integer; Sr: real;
begin
 write('Колькасць навучэнцаў ');
 readln(n);
 writeln('Уводзьце адзнакі');
 for var i := 1 to n do
  read(a[i]);
 S := 0;
 for var i := 1 to n do
  S := S + a[i];
 Sr := S / n;
 if Sr > 7 then
  writeln('Добрая')
 else
  if Sr < 4 then
   writeln('Дрэнная')
  else
   writeln('Сярэдняя');
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць навучэнцаў 5
Уводзьце адзнакі
10 5 6 8 9
Добрыя
```

VII. Аналіз вынікаў.

Прыклад 4.4.
V. Праграма:

```
var Kol, Cen: array[1..50] of
    integer;
  n, Sum: integer;
begin
 write('Увядзіце колькасць
       відаў тавараў ');
 readln(n);
 for var i := 1 to n do
 begin
  writeln('Увядзіце колькасць
      тавару', i, ' і яго цану ');
  read(Kol[i], Cen[i]);
 end;
 Sum := 0;
 for var i := 1 to n do
  Sum := Sum + Kol[i]*Cen[i];
 writeln('Сумарны кошт
         тавараў =', Sum);
end.
```

II. Вынік: адно са слоў — «добрая», «сярэдняя», «дрэнная» ў залежнасці ад значэння сярэдняга бала.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных. Спачатку ўводзім колькасць навучэнцаў у класе, затым масіў адзнак (паэлементна з клавіятуры).

2. Для вызначэння паспяховасці трэба вылічыць сярэдні бал (пераменная Sr). Сярэдні бал вызначаецца як сума (пераменная S) усіх адзнак, падзеленая на колькасць навучэнцаў у класе. Пачатковае значэнне для сумы — S := 0.

3. У цыкле дабаўляем чарговы элемент масіву да сумы.

4. Дзелім атрыманую суму на колькасць навучэнцаў у класе.

5. Правяраем значэнне сярэдняга бала і выводзім вынік.

IV. Апісанне пераменных: a — **array**[1..30] **of** integer; n, S — integer; Sr — real.

4.2. Вылічэнне сум і здабыткаў пры рабоце з двума масівамі

Прыклад 4.4. На складзе захоўваюцца тавары. Для кожнага віду тавару вядома колькасць адзінак тавару і кошт за адзінку тавару. Вызначыць сумарны кошт усіх тавараў, якія захоўваюцца на складзе.

I. Зыходныя даныя: Cen — аднамерны масіў для захоўвання кошту адзінкі тавару кожнага віду, Kol — масіў для захоўвання колькасці тавару кожнага віду, n — колькасць відаў тавараў.

II. Вынік: Sum — значэнне сумарнага кошту тавараў на складзе.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных. Для кожнага віду тавару задаецца яго цана і колькасць.

2. Кошт усіх тавараў аднаго віду вызначаецца як здабытак колькасці на кошт. Сумарны кошт — сума ўсіх такіх здабыткаў. Пачатковае значэнне сумы `Sum := 0`. У цыкле да сумы дадаюцца здабыткі `Kol[i]*Cen[i]`.

3. Вывад выніку.

IV. Апісанне пераменных: `Cen, Kol — array[1..50] of integer; n, Sum — integer`.

4.3*. Выкарыстанне масіву, элементы якога з'яўляюцца канстантамі

Прыклад 4.5. Зададзены натуральны лік `n` (`n < 5000`). Вызначыць, ці з'яўляецца гэты лік простым.

I. Зыходныя даныя: `n` — натуральны лік.

II. Вынік: вывад паведамлення «просты» ці «састаўны».

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Вядома, што лік `n` з'яўляецца простым, калі ён не дзеліцца ні на адзін просты лік, не большы за $\sqrt{n}$. Максімальны лік па ўмове — 5000, $\sqrt{5000} \approx 70,7107$. Створым масіў канстант `s_n` з простых лікаў, не большых за 71.

3. У цыкле будзем дзяліць лік `n` на кожны з лікаў, не большы за $\sqrt{n}$, якія захоўваюцца ў масіве

Прыклад 4.4. *Працяг.*

VI. Тэсціраванне.

Окно вывода
```
Увядзіце колькасць відаў тавараў 3
Увядзіце колькасць тавару 1 і яго цану
5 2
Увядзіце колькасць тавару 2 і яго цану
7 3
Увядзіце колькасць тавару 3 і яго цану
4 5
Сумарны кошт тавараў = 51
```

VII. Аналіз вынікаў.

Прыклад 4.5.

Умова `s_n[i] <= sqrt(n)` правяраецца доўга за кошт выкліку функцыі `sqrt(n)`. Гэту ўмову звычайна замяняюць эквівалентнай: `s_n[i]*s_n[i] <= n`.

V. Праграма:

```
const s_n: array of integer =
(2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
1, 37, 41, 43, 47, 53, 59, 61, 67, 71);
var n,i: integer;
begin
 writeln('Увядзіце лік');
 read(n);
 i := 0;
 while (s_n[i] * s_n[i] <= n)
      and (n mod s_n[i] <> 0) do
  i := i + 1;
 if s_n[i]*s_n[i] > n then
  writeln('Просты')
 else
  writeln('Састаўны')
end.
```

VI. Тэсціраванне.

Окно вывода
```
Увядзіце лік
2027
Просты
```

Окно вывода
```
Увядзіце лік
2021
Састаўны
```

VII. Аналіз вынікаў. Праверыць правільнасць вылічэнняў можна на калькулятары або паглядзець у табліцы простых лікаў[1].

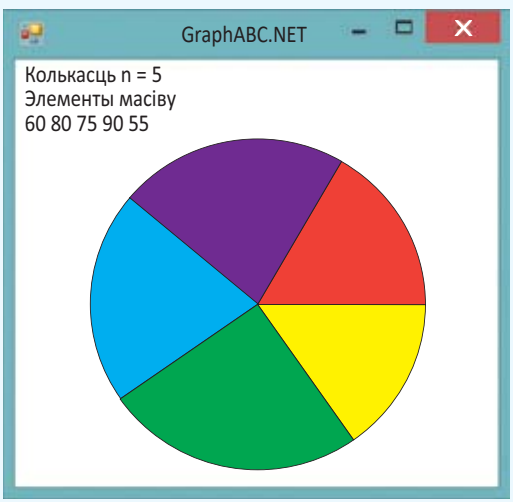
[1] Тэхнічныя табліцы: http://tehtab.ru/guide/guidemathematics/guidemathematicsfigurestables/simplefigures/ (дата доступу: 10.02.2019).

Прыклад 4.6.

V. Праграма:

```
uses graphABC;
var a: array[1..10] of integer;
    n, S, u0, u1: integer;
begin
 write('Колькасць n =');
 readln(n);
 writeln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 for var i := 1 to n do
  write(a[i], ' ');
 S := 0;
 for var i := 1 to n do
  S := S + a[i];
 u0 := 0;
 for var i := 1 to n do
 begin
  u1:= u0+trunc(a[i]*360/S);
  SetBrushColor(clRandom);
  Pie(150,150,100,u0,u1);
  u0 := u1;
 end;
end.
```

VI. Тэсціраванне.



канстант. Калі лік n не падзяліўся ні на адзін з іх, то ён — просты, інакш — састаўны.

4. Правяраем, з якой умовай скончыў работу цыкл: лік з'яўляецца простым, калі апошні прагледжаны элемент масіву большы за $\sqrt{n}$ (лік ні на што не падзяліўся).

5. Вывад выніку.

IV. Апісанне пераменных: s_n — const **array of** integer; n, i — integer.

4.4. Пабудова кругавой дыяграмы

Прыклад 4.6. Зададзены аднамерны масіў з цэлых лікаў. Пабудаваць кругавую дыяграму па лікавых даных, якія захоўваюцца ў масіве. Напрыклад, для 5 элементаў масіву — 60, 80, 75, 90, 55.

I. Зыходныя даныя: масіў a для захоўвання даных і n — колькасць даных.

II. Вынік: кругавая дыяграма.

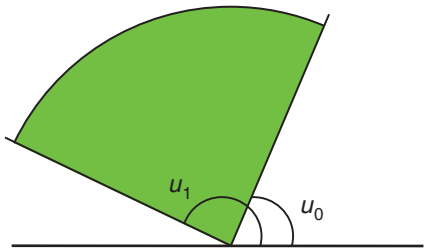
III. Алгарытм рашэння задачы.

1. Увод зыходных даных. Масіў уводзіцца паэлементна з клавіятуры.

2. Кругавая дыяграма складаецца з n сектараў. Градусная мера сектара вызначаецца лікавым значэннем адпаведнага элемента ў масіве. Сумарнае значэнне ўсіх элементаў масіву (пераменная S) адпавядае велічыні ў 360°. Тады значэнню элемента масіву A[i] будзе адпавядаць велічыня — $A[i] * \frac{360}{S}$.

3. Вылічваем суму ўсіх элементаў масіву.

4. У цыкле будуем сектары, градусная мера якіх роўная цэлай частцы велічыні $A[i] * \frac{360}{S}$.



Для пабудовы сектара трэба ведаць велічыні двух вуглоў: u_0 і u_1. Значэнне $u_1 = u_0 + A[i] * \frac{360}{S}$. Спачатку $u_0 = 0$. Затым у цыкле мяняем значэнне u_0 на u_1. Колер сектара будзем задаваць выпадковым чынам. Для вылічэння цэлай часткі можна выкарыстоўваць функцыі `trunc` і `round`.

IV. Апісанне пераменных: `a` — **`array`**`[1..10]` **`of`** `integer`; `n`, `S`, u_0, u_1 – `integer`.

Прыклад 4.6. *Працяг.*

VII. Пабудуйце па гэтых даных дыяграму ў Excel і параўнайце.

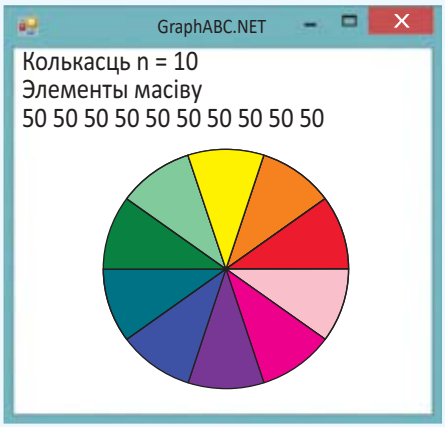
Для задання колеру сектара можна выкарыстоўваць масіў, які змяшчае колеравыя канстанты:

```
const d_color: array [1..10]
of Color = (clRed, clOrange,
clYellow, clLightGreen, clGreen,
clTeal, clBlue, clDarkViolet,
clMagenta, clPink);
```

Каманду задання колеру сектара трэба будзе замяніць на:

```
SetBrushColor(d_color[i]);
```

Вынік:



1. Якія аперацыі дапушчальныя для элементаў масіву цэлых лікаў?
2. Якія аперацыі дапушчальныя для элементаў масіву рэчыўных лікаў?
3. Як запісаць даныя ў масіў канстант?

Практыкаванні

1 Для задачы з прыкладу 4.2 выканайце пералічаныя заданні.

1. Запоўніце табліцу.

	n	a	S	P
1	3	−2 −3 −5		
2	5	1 2 3 4 5		
3	10	1 −3 −2 3 4 3 2 4 3 2		

2. Дабаўце ў табліцу свае значэнні n і a.
3. Паспрабуйце падабраць такія значэнні элементаў масіву, каб S = P, для n = 2,5.
4. Для n = 10 узялі ўсе элементы масіву, роўныя 9. Які вынік атрымалі? Чаму? Што трэба выправіць у праграме для атрымання правільнага выніку?

2 Для задачы з прыкладу 4.3 дабаўце вывад сярэдняга бала.

3 У ходзе хакейнага матча выдаляліся гульцы абедзвюх каманд. Для кожнага выдаленага гульца вядомы час яго адсутнасці на полі. Вызначыце, якая з каманд правяла больш часу на лаўцы штрафнікоў.

4 Для задачы з прыкладу 4.5 выканайце наступнае заданне:

Увядзіце лік 5557. Чаму з'явілася памылка? Дапоўніце масіў канстант простымі лікамі так, каб праграма магла выдаваць адказ для лікаў, меньшых за 10 000. (Для гэтага можна выкарыстаць саму праграму або табліцу простых лікаў.)

5 Для задачы з прыкладу 4.6 выканайце пералічаныя заданні.

1. Унясіце ў праграму змяненні так, каб колер сектара выбіраўся з масіву канстант.
2*. Змяніце праграму так, каб дыяграма заўсёды будавалася ў цэнтры графічнага акна. Дыяметр круга вызначаецца меншай з дзвюх велічынь — шырынёй або вышынёй акна.

6* У масівах X і Y захоўваюцца каардынаты пунктаў. Пабудуйце многавугольнік, зададзены гэтымі каардынатамі. Запытайце ў карыстальніка нумары двух пунктаў і пабудуйце дыяганаль многавугольніка, якая злучае гэтыя пункты.

§ 5. Пошук элементаў з зададзенымі ўласцівасцямі

Чалавек увесь час сутыкаецца з задачамі пошуку патрэбнай інфармацыі. Тыповым прыкладам можа служыць праца з даведнікамі ці бібліятэчнай картатэкай. У сучасным свеце інфармацыю шукаюць з выкарыстаннем сеткі Інтэрнэт.

Каб пошук быў выніковым і хуткім, распрацоўваюць эфектыўныя алгарытмы пошуку. Важную ролю ў працэсе пошуку інфармацыі адыгрывае спосаб захоўвання даных. Адной з самых простых структур для гэтага з'яўляецца масіў.

5.1. Лінейны пошук

Разгледзім, як ажыццяўляецца пошук для даных, што захоўваюцца ў масіве.

Сярод разнавіднасцей найпрасцейшых задач пошуку, якія сустракаюцца на практыцы, можна вылучыць наступныя тыпы:

1. Знайсці хоць бы адзін элемент, роўны зададзенаму элементу X. У выніку неабходна атрымаць i — індэкс

(нумар) элемента масіву, такі, што a[i] = X.

2. Знайсці ўсе элементы, роўныя зададзенаму X. У выніку неабходна атрымаць колькасць такіх элементаў і (ці) іх індэксы.

Часам пошук арганізуецца не па супадзенні з элементам X, а па выкананні некаторых умоў. Прыкладам можа служыць пошук элементаў, што задавальняюць умову: $X_1 \leqslant a[i] \leqslant X_2$, дзе X_1 і X_2 зададзены.

Калі няма ніякай дадатковай інфармацыі пра даныя, якія трэба знайсці, то самы просты падыход — паслядоўны прагляд элементаў масіву.

Алгарытм, пры якім для пошуку патрэбнага элемента паслядоўна праглядаюць усе элементы масіву ў парадку іх запісу, называецца **лінейным** або **паслядоўным пошукам.**

5.2. Пошук аднаго элемента, які задавальняе ўмову пошуку

Прыклад 5.1. Зададзены аднамерны масіў з n лікаў. Вызначыць, ці ёсць у ім хоць бы адзін элемент, роўны x (значэнне x уводзіцца).

I. Зыходныя даныя: масіў a, колькасць лікаў n, шуканы лік x.

II. Вынік: вывад паведамлення «Элемент знойдзены» або «Элемент не знойдзены».

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Няхай p — пераменная лагічнага тыпу, якая мае значэнне «праўда», калі элемент у масіве

Алгарытмы пошуку можна падзяліць на алгарытмы, якія выкарыстоўваюць неўпарадкаваныя наборы даных, і на алгарытмы, што працуюць з папярэдне ўпарадкаваным наборам даных.

Прыкладам пошуку ў неўпарадкаваным наборы даных можа служыць пошук сшытка пэўнага навучэнца ў стосе сшыткаў, зададзеных на праверку. Каб знайсці патрэбны сшытак, магчыма, прыйдзецца перагледзець усё. Пошук у слоўніку — пошук ва ўпарадкаваным наборы даных, паколькі ўсе словы размешчаны ў алфавітным парадку.

Прыклад 5.1.
V. Праграма:

```
var a: array[1..10] of integer;
 n, x: integer;
 p: boolean;
begin
 write('Колькасць n =');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 write('Лік x =');
 readln(x);
 //лінейны пошук элемента
 p := false;
 for var i := 1 to n do
  if a[i] = x then
   p := true;
 if p then
  writeln('Элемент знойдзены')
 else
  writeln('Элемент не знойдзены');
end.
```

VI. Тэсціраванне.

Окно вывода
Колькасць n = 5 Элементы масіву 1 2 3 4 5 Лік x = 4 Элемент знойдзены

Окно вывода
Колькасць n = 5 Элементы масіву 1 3 5 7 9 Лік x = 6 Элемент не знойдзены

Прыклад 5.2.

V. Праграма:

```
var a: array[1..10] of integer;
 n, x, k: integer;
begin
 write('Колькасць n =');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 write('Лік x =');
 readln(x);
 //лінейны пошук элемента
 k := 0;
 for var i := 1 to n do
  if a[i] = x then
   k := i;
 if k = 0 then
  writeln('Элемент не знойдзены')
 else
  writeln('Элемент знойдзены
          на месцы ', k);
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 5
Элементы масіву
1 3 3 3 5
Лік x = 3
Элемент знойдзены
на месцы 4
```

Окно вывода

```
Колькасць n = 5
Элементы масіву
1 3 5 7 9
Лік x = 4
Элемент не знойдзены
```

Прыклад 5.3.
Фрагмент праграмы:

```
//лінейны пошук элемента
 k := 1;
 while (k<=n) and (a[k]<>x) do
  k := k + 1;
 if k = n + 1 then
  writeln('Элемент не знойдзены')
 else
  writeln('Элемент знойдзены
          на месцы ', k);.
```

знойдзены, і «няпраўда» — у адваротным выпадку. Да прагляду элементаў масіву p := false.

3. У цыкле будзем праглядаць усе лікі ў масіве і параўноўваць іх з лікам x.

4. Пасля заканчэння пошуку магчымая адна з дзвюх сітуацый:

4.1. Элемент знойдзены (p := true), г. зн. у масіве ёсць такі элемент a[i], што a[i] = x.

4.2. Увесь масіў прагледжаны, і супадзення не выяўлена (p := false).

5. Вывад выніку.

IV. Апісанне пераменных: a — **array**[1..10] **of** integer; n, x — integer; p: boolean.

Часта патрабуецца не толькі вызначыць, ці ёсць у масіве шуканы элемент, але і знайсці, на якім месцы ён знаходзіцца.

Будзем захоўваць індэкс знойдзенага элемента (прыклад 5.2) у пераменнай k. Пасля выканання дадзенага алгарытму па значэнні пераменнай k можна вызначыць, ці ёсць у масіве шуканы элемент, і калі ёсць, то дзе ён стаіць. Калі ў масіве некалькі такіх элементаў, то ў пераменнай k будзе захоўвацца нумар апошняга з іх. Калі такога элемента няма, то значэнне пераменнай k не зменіцца (k застанецца роўным 0).

На практыцы аперацыю пошуку даводзіцца выконваць дастаткова часта, і хуткасць работы праграмы знаходзіцца ў прамой залежнасці ад алгарытму пошуку, што выкарыстоўваецца.

У разгледжаных вышэй алгарытмах патрабуецца прагледзець увесь масіў, нават у тым выпадку, калі шуканы элемент знаходзіцца ў масіве на першым месцы.

Для скарачэння часу пошуку можна спыняцца адразу пасля таго, як элемент знойдзены. У гэтым выпадку ўвесь масіў прыйдзецца прагледзець толькі тады, калі шуканы элемент апошні ці яго няма наогул (прыклад 5.3). Цыкл заканчвае работу, калі будзе знойдзены шуканы элемент або калі k = n + 1, г. зн. элемента, які супадае з x, не існуе.

*Пры такой рэалізацыі на кожнай ітэрацыі цыкла патрабуецца павялічваць індэкс k і вылічаць лагічны выраз. Паскорыць пошук можна, спрасціўшы лагічны выраз. Змесцім у канец масіву дадатковы элемент са значэннем x. Тады супадзенне з x абавязкова адбудзецца, і можна не правяраць умову a[k] < > x. Такі дапаможны элемент часта называюць **«бар'ерам»** або **«вартавым»**, паколькі ён перашкаджае выхаду за межы масіву. У зыходным масіве цяпер будзе n + 1 элемент (прыклад 5.4).

5.3. Знаходжанне ўсіх элементаў, якія задавальняюць умову пошуку

Калі патрабуецца вызначыць колькасць элементаў, якія задавальняюць якую-небудзь умову, то для гэтага вызначаюць асобную пераменную, значэнне якой павялічваюць на 1 кожны раз, калі знойдзены патрэбны элемент. Такую пераменную называюць

Прыклад 5.4*.
Фрагмент праграмы:

```
//линейный поиск с барьером
 a[n + 1] := x;
 k := 1;
 while a[k]<>x do
  k := k + 1;
 if k = n + 1 then
  writeln('Элемент не знойдзены')
 else
  writeln('Элемент знойдзены
          на месцы ', k);
```

V. Тэсціраванне.

Окно вывода	Окно вывода
Колькасць n = 5 Элементы масіву 1 5 6 7 1 Лік x = 7 Элемент знойдзены на месцы 4	Колькасць n = 5 Элементы масіву 1 2 3 4 5 Лік x = -2 Элемент не знойдзены

У сучасных мовах праграмірования выкарыстоўваюцца бібліятэкі, якія змяшчаюць функцыі для пошуку элементаў у масівах (і іншых структурах даных). У PascalABC.Net такія функцыі рэалізаваны толькі для дынамічных масіваў (памер масіву можа змяняцца ў час выканання праграмы). Апісанне функцый можна знайсці ў даведніку ў раздзеле «Метады пашырэння аднамерных дынамічных масіваў». Прыклад выкарыстання функцыі пошуку ўсіх элементаў масіву, большых за 5:

```
var c : array of integer;
begin
 setlength(c, 10);
 for var i := 0 to 9 do
  c[i] := random(1, 10);
 println(c);
 c.FindAll(p -> p > 5). Println;
end.
```

Прыклад 5.5.
V. Праграма:

```
var a: array[1..10] of integer;
  n, x, k: integer;
begin
  write('Колькасць n =');
  readln(n);
  writeln('Элементы масіву');
  for var i := 1 to n do
    read(a[i]);
  write('Лік x =');
  readln(x); k := 0;
  for var i := 1 to n do
    if a[i] mod x = 0 then
      k := k + 1;
  writeln('У масіве ', k, '
    элемент(-ы,-аў), кратны(-х)', x);
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 5
Элементы масіву
1 2 3 4 5
Лік x = 2
У масіве 2 элемент(-ы,-аў),
кратны(-х) 2
```

VII. Аналіз вынікаў.

Прыклад 5.6.
V. Праграма:

```
var a, b: array[1..10] of integer;
  n, x, k: integer;
begin
  write('Колькасць n =');
  readln(n);
  writeln('Элементы масіву');
  for var i := 1 to n do
    read(a[i]);
  write('Лік x =');
  readln(x); k := 0;
  for var i := 1 to n do
    if a[i] mod x = 0 then
    begin
      k := k + 1; b[k] := i;
    end;
  writeln('У масіве ', k, '
    элемент(-ы,-аў), кратны(-х) ', x);
  writeln('Месцазнаходжанне ');
  for var i := 1 to k do
    write(b[i],' ');
end.
```

лічыльнікам. Да пачатку прагляду элементаў масіву лічыльніку трэба задаць пачатковае значэнне, ці, іншымі словамі, **ініцыялізаваць** значэнне пераменнай. У выпадку падліку колькасці элементаў, якія задавальняюць умову, лічыльнік ініцыялізуецца нулём. Для рашэння задачы трэба праглядаць увесь масіў.

Прыклад 5.5. Зададзены аднамерны масіў з n лікаў. Вызначыць колькасць элементаў, кратных x, у лінейным масіве.

I. Зыходныя даныя: масіў a, колькасць лікаў n, шуканы лік x.

II. Вынік: колькасць элементаў, якія задавальняюць умову, — k.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. Ініцыялізацыя лічыльніка.
3. У цыкле будзем праглядаць усе лікі ў масіве і параўноўваць з нулём іх астачы ад дзялення на лік x. Калі астача роўна нулю, то лічыльнік павялічваем на 1.
4. Вывад выніку.

IV. Апісанне пераменных: a — **array**[1..10] **of** integer; n, x, k — integer.

Калі неабходна не толькі палічыць, колькі элементаў задавальняюць умову, але і захаваць індэксы такіх элементаў, то для гэтага можна выкарыстаць дадатковы масіў. Створым новы масіў b. Як толькі будзе знойдзены неабходны элемент, яго індэкс будзе заносіцца ў масіў b. Пераменная k будзе захоўваць нумар апошняга занятага месца ў масіве b. Спачатку k = 0 (прыклад 5.6).

Пасля завяршэння работы першыя k элементаў масіву b будуць змяшчаць індэксы шуканых элементаў.

Калі для рашэння задачы спатрэбяцца значэнні ўсіх знойдзеных элементаў, то ў праграме магчымы такі зварот да элементаў масіву: a[b[i]]. Адрас элемента ў масіве a будзе вызначацца значэннем элемента масіву b па адрасе i. Для выведу значэнняў элементаў у прыкладзе 5.6 апошні цыкл трэба замяніць на

```
for var i := 1 to k do
  write(a[b[i]],' ');
```

5.4. Рашэнне задач з выкарыстаннем алгарытму лінейнага пошуку

Прыклад 5.7. Вядомыя вынікі ЦТ па матэматыцы для n чалавек. Вызначыць, ці ёсць сярод іх хоць бы адзін чалавек з балам, вышэйшым за x. Значэнне x уводзіцца з клавіятуры. Вынікі экзамену атрымаць выпадковым чынам.

I. Зыходныя даныя: масіў a, колькасць лікаў n, лік x.

II. Вынік: паведамленне адпавядае ўмове задачы.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Прагляд элементаў з пачатку. Як толькі элемент знойдзены, спынімся.

3. Калі ўвесь масіў прагледжаны, значыць, у зыходным масіве няма элемента, які задавальняе ўмову задачы, інакш выводзім нумар знойдзенага элемента.

IV. Апісанне пераменных: a — **array**[1..20] **of** integer; n, x, k — integer.

Прыклад 5.6. *Працяг.*

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 5
Элементы масіву
6 3 2 4 5
Лік x = 2
У масіве 3 элемент(-ы,-аў),
кратны(-х) 2
Месцазнаходжанне
1 3 4
```

Прыклад 5.7.

V. Праграма:

```
var a: array[1..20] of integer;
 n, x, k: integer;
begin
 write('Колькасць n =');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
 begin
  a[i] := random(0,100);
  write(a[i],' ');
 end;
 writeln;
 write('Лік x =');
 readln(x);
 //лінейны пошук з бар'ерам
 a[n+1] := x + 1;
 k := 1;
 while (a[k] <= x) do
  k := k + 1;
 if k = n+1 then
  writeln('Няма такіх')
 else
  writeln('Гэта чалавек з № ',
          k, ', яго бал — ', a[k]);
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 10
Элементы масіву
48 12 96 48 9 95 5 71 77 24
Лік x = 80
Гэта чалавек з №3, яго бал - 96
```

Прыклад 5.8.
V. Праграма:

```
uses graphABC;
var X,Y: array [1..1000] of
    integer;
  n, k1, k2, R: integer;
begin
 write('Колькасць пунктаў  n =');
 read(n);
 writeln(n);
 for var i := 1 to n do
 begin
  X[i]:= random(-200,200);
  Y[i]:= random(-200,200);
 end;
 writeln('Радыус акружнасці');
 read(R);
 writeln(R);
 {Пабудова акружнасці і
 восей каардынат}
 circle(200,200,R);
 line(0,200,400,200);
 line(200,0,200,400);
 k1 := 0; k2 := 0;
 for var i := 1 to n do
  if X[i]*X[i]+Y[i]*Y[i]<=R*R then
  begin
   k1 := k1 + 1;
   SetPixel(X[i]+200,200-Y[i],
            clred);
 end
 else
 begin
  k2 := k2 + 1;
   SetPixel(X[i]+200,200-Y[i],
            clblue);
 end;
 if k1 > k2 then
  writeln('Унутры больш')
 else
 if k1<k2 then
   writeln('Звонку больш')
  else
   writeln('Пароўну')
end.
```

Прыклад 5.8. У двух лінейных масівах X і Y, зададзеных выпадковым чынам, захоўваюцца каардынаты пунктаў плоскасці (–200 ⩽ X[i], Y[i] ⩽ 200). Вызначыць, якіх пунктаў больш — якія ляжаць унутры або звонку вобласці, абмежаванай акружнасцю радыуса R з цэнтрам у пачатку каардынат (будзем лічыць, што пункты, якія ляжаць на акружнасці, ляжаць унутры вобласці). Пабудаваць акружнасць і пункты. Пункты, якія належаць унутранай вобласці, намаляваць чырвоным колерам, а знешняй вобласці — сінім колерам.

I. Зыходныя даныя: X, Y — масівы лікаў, R — радыус акружнасці, n — колькасць пунктаў.

II. Вынік: малюнак, які адпавядае ўмове задачы, і паведамленне: «Унутры пунктаў больш», «Звонку пунктаў больш» ці «Пунктаў пароўну».

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Ініцыялізацыя лічыльнікаў: k1:= 0; k2:= 0.

3. Будзем праглядаць усе пункты і для кожнага правяраць прыналежнасць вобласці. Калі $X^2 + Y^2 \leqslant R^2$, то пункт ляжыць унутры вобласці, тады павялічым значэнне лічыльніка k1 на 1, калі не, то павялічым на 1 значэнне лічыльніка k2.

4. Параўнаем значэнні k1 і k2 і выведзем вынік.

5. Паколькі каардынаты пунктаў належаць адрэзку [–200, 200], то восі каардынат можна намаляваць перасякальнымі ў пункце з каардынатамі (200, 200). Для

пераўтварэння каардынат пунктаў у экранныя трэба да значэння абсцысы прыбавіць 200, а значэнне ардынаты трэба адняць ад 200 (вось Y на экране накіравана ўніз, таму трэба памяняць знак ардынаты). Будаваць пункты можна ў тым жа цыкле, у якім адбываецца праверка.

IV. Апісанне пераменных: X, Y — `array[1..1000] of integer`; n, k1, k2, R — `integer`.

Прыклад 5.9. На складзе захоўваюцца пустыя скрыні для ўпакоўвання тавару. Вядомая, што маса аднаго пакета з цукеркамі x кг. Якая сумарная маса пакетаў з цукеркамі, якія можна ўпакаваць у такія скрыні, запоўніўшы скрыні цалкам?

I. Зыходныя даныя: масіў a, колькасць лікаў n, лік x.

II. Вынік: колькасць скрынь — k, сумарная маса — S.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Ініцыялізацыя лічыльніка і значэння сумы: `k := 0; S := 0`.

3. Праглядаючы масіў, праверым, ці з'яўляецца бягучы элемент лікам, кратным x (у гэтым выпадку скрыня будзе запоўнена цалкам). Як толькі элемент знойдзены, павялічым лічыльнік k на 1, а пераменную *S* на значэнне знойдзенага элемента масіву.

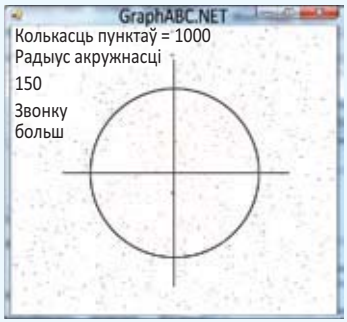
4. Вывад выніку.

IV. Апісанне пераменных: a — `array[1..20] of integer`; n, x, k, S — `integer`.

Прыклад 5.10. Маецца спіс хлопчыкаў 10 В класа і вынікі іх бегу на

Прыклад 5.8. *Працяг.*

VI. Тэсціраванне.



Прыклад 5.9.

V. Праграма:

```
var a: array [1..20] of integer;
    n, x, k, S: integer;
begin
 write('Колькасць скрынь:');
 readln(n);
 writeln('Умяшчальнасць
         скрынь');
 for var i := 1 to n do
  read(a[i]);
 write('Маса цукерак:'); read(x);
 k := 0; S := 0;
 for var i := 1 to n do
  if a[i] mod x = 0 then
  begin
     k := k + 1;
     S := S + a[i];
  end;
 writeln('На складзе',k,'скр.');
 writeln('Сумарная маса', S);
end.
```

VI. Тэсціраванне.

```
Окно вывода
Колькасць скрынь: 8
Умяшчальнасць скрынь
15 23 64 27 35 10 48 13
Маса цукерак: 5
На складзе 3 скрынь(-i,-яў)
Сумарная маса: 60
```

VII. Аналіз вынікаў. Скрыні, якія задавальняюць умову задачы, маюць вагу — 15, 25 і 10.

Прыклад 5.10.

V. Праграма:

```
var r: array [1..20] of real;
  fam: array [1..20] of string;
  n, k: integer;
begin
 writeln('Колькасць навучэнцаў: ');
 readln(n);
 writeln('Прозвішча і вынік: ');
 for var i := 1 to n do
 begin
   readln(fam[i]);
   readln(r[i]);
 end;
 writeln('Прозвішчы тых, хто не
          здаў нарматыў:');
 k := 0;
 for var i := 1 to n do
  if r[i] > 16 then
  begin
    k := k + 1;
    writeln(fam[i]);
  end;
 writeln('Не здалі нарматыў:', k);
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць навучэнцаў:
5
Прозвішча і вынік:
Іваноў
13.5
Пятроў
14.0
Сідараў
21
Каралёў
16.1
Верамей
15.9
Прозвішчы тых, кто не здаў нарматыў:
Сідараў
Каралёў
Не здалі нарматыў: 2
```

VII. Аналіз вынікаў. Вынік Сідарава — 21, а Каралёва — 16.1, што перавышае нарматыў.

100 м. Для здачы нарматыву неабходна прабегчы дыстанцыю не больш чым за 16 с. Вывесці прозвішчы навучэнцаў, якія не выканалі нарматыў па бегу. Колькі такіх навучэнцаў у класе?

I. Зыходныя даныя: масівы `fam` (прозвішчы навучэнцаў) і `r` (вынікі бегу ў секундах), колькасць навучэнцаў `n`.

II. Вынік: прозвішчы тых навучэнцаў, якія не выканалі нарматыў па бегу.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Ініцыялізацыя лічыльніка: `k := 0`.

3. Будзем праглядаць масіў з вынікамі і правяраць, ці з'яўляецца бягучы элемент лікам, большым за 16 (нарматыў не здадзены). Калі такое значэнне знойдзена, то выведзем элемент масіву `fam` з адпаведным нумарам і павялічым значэнне лічыльніка на 1.

4. Вывад значэння лічыльніка.

IV. Апісанне пераменных: `fam` — **array**[1..20] **of** string; `r` — **array**[1..20] **of** real, `n`, `k` — integer.

Прыклад 5.11*. Зададзены аднамерны масіў з `N` цэлых лікаў. Вызначыць колькасці элементаў, якія з'яўляюцца лікамі Сміта. (Лік Сміта — гэта такі састаўны лік, сума лічбаў якога роўна суме лічбаў усіх яго простых сумножнікаў.) Напрыклад, лікам Сміта з'яўляецца $202 = 2 \times 101$, паколькі $2+0+2=4$ і $2+1+0+1=4$.

I. Зыходныя даныя: `a` — масіў лікаў, `n` — колькасць лікаў у масіве.

II. Вынік: лікі Сміта і іх колькасць у масіве.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Ініцыялізацыя лічыльніка: k := 0.

3. Будзем праглядаць кожны элемент масіву і вызначаць, ці з'яўляецца ён лікам Сміта. Для праверкі створым функцыю check, якая будзе атрымліваць у якасці параметра элемент масіву, а таксама вяртаць значэнне true, калі лік з'яўляецца лікам Сміта, і false ў адваротным выпадку.

3.1. Знойдзем суму лічбаў ліку.

3.2. Будзем раскладаць лік на простыя множнікі і для кожнага множніка знаходзіць суму лічбаў.

3.3. Для раскладання ліку на простыя множнікі будзем дзяліць яго спачатку на 2 (пакуль дзеліцца), затым на 3. На 4 лік ужо дзяліцца не будзе, будзем дзяліць яго на 5 і г. д. Скончыцца раскладанне тады, калі пасля ўсіх дзяленняў лік стане роўным 1.

4. Таксама нам спатрэбіцца функцыя sum, якая для ліку будзе вылічаць яго суму лічбаў.

IV. Апісанне пераменных: a — **array**[1..100] **of** integer; n, k — integer.

Прыклад 5.12*. Зададзены аднамерны масіў з n радкоў. Кожны радок з'яўляецца сказам са слоў, падзеленых прабеламі. Знайсці і вывесці тыя сказы, у якіх няцотная колькасць слоў.

Прыклад 5.11*.

V. Праграма:

```
var a: array [1..100] of
   integer;
   n, k: integer;

function sum(x: integer):
         integer;
var s: integer;
begin
 s := 0;
 while x > 0 do
 begin
  s := s + x mod 10; x := x div 10;
 end;
 sum := s;
end;

function check(x: integer):
         boolean;
var s1, s2, d: integer;
begin
 s1 := sum(x); s2 := 0; d := 2;
//раскладанне на простыя множнікі
 while x <> 1 do
 begin
  while x mod d = 0 do
  begin
   s2 := s2 + sum(d);
   x := x div d;
  end;
  d := d + 1;
 end;
 check := s1 = s2;
end;
begin
 writeln('Колькасць');
 readln(n);
 writeln('Элементы');
 for var i := 1 to n do
  read(a[i]);
 k := 0;
 writeln('Лікі Сміта');
 for var i := 1 to n do
  if check(a[i]) then
  begin
   inc(k);
   write(A[i], ' ');
  end;
 writeln;
 writeln('Усяго - ', k);
end.
```

Прыклад 5.11*. *Працяг.*
VI. Тэсціраванне.

Окно вывода
Колькасць 5 Элементы 202 3 323 85 117 Лік Сміта 202 3 85 Усяго - 3

Окно вывода
Колькасць 5 Элементы 8 8 8 8 8 Лік Сміта Усяго - 0

Прыклад 5.12*.
V. Праграма:

```
var a: array [1..100] of
   string;
   n, k: integer;

function check(x: string):
         integer;
var
 s, len: integer;
begin
 s := 0;
 x := ' ' + x;
 len := length(x);
 for var i := 1 to len - 1 do
  if (x[i] = ' ') and (x[i + 1] <> ' ')
  then
    inc(s);
 check := s;
end;

begin
 writeln('Колькасць');
 readln(n);
 writeln('Элементы');
 for var i := 1 to n do
  readln(a[i]);
 k := 0;
 writeln;
 writeln('Шуканыя радкі:');
 for var i := 1 to n do
  if check(a[i]) mod 2 <> 0 then
  begin
   inc(k);
   writeln(a[i]);
  end;
 writeln('Усяго - ', k);
end.
```

I. Зыходныя даныя: `a` — масіў радкоў, `n` — колькасць радкоў у масіве.

II. Вынік: шуканыя радкі і іх колькасць.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Ініцыялізацыя лічыльніка: `k := 0;`

3. Будзем праглядаць кожны радок і вызначаць, колькі ў ім слоў. Для праверкі створым функцыю `check`, якая будзе атрымліваць у якасці параметра элемент масіву і вылічаць колькасць слоў у радку. Калі колькасць слоў з'яўляецца нецотным лікам, то выведзем радок і павялічым значэнне лічыльніка.

3.1. Перад кожным словам сказа, акрамя першага, стаіць прабел, слова пачынаецца з сімвала, які прабелам не з'яўляецца.

3.2. Дабавім прабел перад першым словам, тады колькасць слоў будзе вызначацца колькасцю спалучэнняў пар сімвалаў: прабел і не прабел.

Апісанне пераменных: `a` — **array**`[1..100]` **of** `string; n, k` — `integer`.

Пры ўводзе даных для тэсціравання праграмы трэба памятаць, што пасля кожнага сказа неабходна націскаць клавішу Enter.

У дадзеным выпадку радок як тып даных можа не адпавядаць радку ў акне выведу. У прыкладзе 5.12 уводзяцца 3 радкі:

1. Шмат якія кампаніі распрацоўваюць свае правілы па афармленні кода.

2. У іх прапісаны таксама правілы называння пераменных.

3. Кампанія Microsoft выкарыстоўвае так званую «венгерскую натацыю».

У акне выведу колькасць радкоў можа быць большай (на малюнку іх 6).

То, як будуць выглядаць уводныя радкі ў акне выведу, залежыць ад шырыні акна дадатка PascalABC.NET. На вялікім маніторы, калі акно дадатка разгорнута на ўвесь экран, радкоў можа быць тры.

Кампілятар вызначае, што ўвод радка скончаны, калі была націснута клавіша Enter. Знешні выгляд радкоў у акне выведу для кампілятара не мае значэння.

Прыклад 5.12*. *Працяг.*

VI. Тэсціраванне.

Окно вывода

```
Колькасць
3
Элементы
Шмат якія кампаніі распрацоўваюць
свае правілы афармлення кода.
У іх прапісаны таксама правілы
называння пераменных.
Кампанія Microsoft выкарыстоўвае так
званую «венгерскую натацыю».

Шуканыя радкі
У іх прапісаны таксама правілы
называння пераменных.
Кампанія Microsoft выкарыстоўвае так
званую «венгерскую натацыю».

Усяго - 2
```

1. Што называюць паслядоўным пошукам?
2. Як вызначыць, што ў масіве быў знойдзены элемент з пэўнымі ўласцівасцямі?
3. Для чаго выкарыстоўваюць пераменныя «лічыльнікі»?
4. Што такое ініцыялізацыя пераменнай?

Практыкаванні

1 Для прыкладаў 5.1—5.3 выканайце пералічаныя заданні.

1. Запоўніце табліцу.

№	n	Масіў	x	Вынік
1	5	2 14 7 20 16	20	
2	5	2 3 4 5 6	8	
3	7	2 4 6 8 10 11 12	8	
4	5	6 3 9 12 15	3	

2. Дабаўце ў табліцу свае такія даныя, каб шуканы элемент быў першым у масіве; апошнім у масіве.

3. Які адказ выдасць кожная з праграм, калі ў масіве некалькі элементаў, якія задавальняюць умову задачы? Чаму?

4. Што трэба змяніць у праграме 5.2, каб выдаваўся не апошні са знойдзеных элементаў, а першы?

5. Што трэба змяніць у праграме 5.1, каб выдаваўся не апошні са знойдзеных элементаў, а першы?

6. Змяніце ўмову цыкла **while** з прыкладу 5.3 так, каб выкарыстоўвалася лагічная аперацыя not.

2 Рост навучэнцаў класа пададзены ў выглядзе масіву. Вызначыце колькасць навучэнцаў, рост якіх большы за сярэдні рост па класе.

3 Зададзены прозвішчы і рост навучэнцаў 10-га класа. Вывесці прозвішчы тых навучэнцаў, рост якіх меншы за сярэдні рост па класе.

4 Вядомыя даныя пра плошчу n краін (у млн кв. км) і колькасць насельніцтва (у млн жыхароў). Выведзіце нумары тых краін, шчыльнасць насельніцтва якіх большая за X.

5 Для практыкавання 4 дабаўце магчымасць уводзіць і выводзіць назвы краін.

6 Вызначыце, ці ёсць у лінейным масіве хоць бы адзін элемент, які з'яўляецца няцотным лікам, кратным 7. Калі так, то трэба вывесці яго нумар.

7* У лінейным масіве знайдзіце і выведзіце ўсе простыя лікі з няцотнай сумай лічбаў. Запішыце, колькі лікаў вывелі.

8* У лінейным масіве знайдзіце і выведзіце ўсе лікі Армстранга. (Лікам Армстранга называецца такі лік, які роўны суме сваіх лічбаў, узведзеных у ступень, роўную колькасці яго лічбаў. Напрыклад, лікам Армстранга з'яўляецца лік 371 : $371 = 3^3 + 7^3 + 1^3 = 27 + 343 + 1$.) Запішыце, колькі лікаў вывелі.

9 Зададзены аднамерны масіў з N радкоў. Кожны радок з'яўляецца сказам са слоў, падзеленых прабеламі. Знайдзіце і выведзіце тыя сказы, у якіх ёсць словы, што пачынаюцца на галосную (малую або вялікую).

§ 6. Максімальны і мінімальны элементы масіву

Прыклад 6.1.
V. Праграма:

```
var a: array[1..20] of integer;
 n, max: integer;
begin
 write('Колькасць n=');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 max := a[1];
 for var i := 2 to n do
  if a[i] > max then
   max := a[i];
 writeln('Максімум = ', max);
end.
```

6.1. Пошук максімальнага (мінімальнага) элемента ў масіве

Вельмі часта для рашэння задачы патрабуецца знаходзіць не зададзены элемент масіву, а максімальны (найбольшы) ці мінімальны (найменшы).

Разгледзім задачу знаходжання максімальнага элемента. Калі ў масіве толькі адзін элемент, то ён і ёсць максімальны. Калі элементаў больш за адзін, то максімальным у масіве з i элементаў з'яўляецца максімум з a[i] і максімальнага сярод першых

i-1 элементаў. Знаходзіць максімум будзем паслядоўна, параўноўваючы бягучы элемент з максімумам, знойдзеным на папярэднім кроку. Калі бягучы элемент большы, то значэнне максімуму, знойдзенае на папярэднім кроку, трэба абнавіць (прыклад 6.1).

Дадзены алгарытм знаходзіць значэнне максімальнага элемента, але не дазваляе вызначыць, на якім месцы ў масіве размешчаны гэты максімальны элемент.

Будзем выкарыстоўваць пераменную n_max для захоўвання індэкса максімальнага элемента. Значэнне пераменнай n_max будзе змяняцца тады, калі змяняецца значэнне максімальнага элемента (прыклад 6.2).

Калі ў масіве некалькі элементаў маюць максімальнае значэнне, то значэннем пераменнай n_max будзе індэкс першага з іх. Калі выкарыстоўваць умову a[i] ⩾ max, то пераменная n_max будзе захоўваць індэкс апошняга з максімальных элементаў.

У выпадку, калі вядомы індэкс i элемента масіву, значэнне элемента можна атрымаць, звярнуўшыся да элемента па індэксе: a[i]. Таму пры пошуку максімальнага элемента дастаткова захоўваць толькі яго індэкс n_max. Значэнне максімальнага элемента — a[n_max] (прыклад 6.3).

Для пошуку мінімальнага элемента неабходна замяніць знак > ва ўмове аператара галінавання на знак < (прыклад 6.4).

Прыклад 6.1. *Працяг.*
VI. Тэсціраванне.

```
Окно вывода
Колькасць n = 5
Элементы масіву
1 2 6 3 2
Максімум = 6
```

Прыклад 6.2.
V. Праграма:

```
var a: array[1..20] of integer;
    n, max, n_max: integer;
begin
 write('Колькасць n =');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 max := a[1]; n_max := 1;
 for var i := 2 to n do
  if a[i] > max then
  begin
   max := a[i]; n_max :=i;
  end;
 writeln('Максімум = ', max);
 writeln('Яго месца ', n_max);
end.
```

VI. Тэсціраванне.

```
Окно вывода
Колькасць n = 5
Элементы масіву
2 5 3 5 1
Максімум = 5
Яго месца 2
```

Прыклад 6.3. Фрагмент праграмы:

```
n_max := 1;
for var i := 2 to n do
 if a[i] > a[n_max] then
  n_max := i;
writeln('Максімум =',a[n_max]);
writeln('Яго месца ', n_max);
```

Прыклад 6.4. Фрагмент праграмы:

```
n_min := 1;
for var i := 2 to n do
 if a[i] < a[n_min] then
  n_min := i;
```

Прыклад 6.5.
V. Праграма:

```
var a: array [1..20] of real;
    n, n_min: integer;
begin
 writeln('Колькасць
        спартсменаў');
 readln(n); writeln('Час');
 for var i := 1 to n do
  read(a[i]);
 //пошук мінімальнага элемента
 n_min := 1;
 for var i := 2 to n do
  if a[i] < a[n_min] then
    n_min := i;
 writeln('Пераможца — лыжнік
         нумар ', n_min);
 writeln('Яго час - ', a[n_min]);
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць спартсменаў
5
Час
6.31 6.17 7.32 6.54 7.03
Пераможца - лыжнік нумар 2
Яго час - 6.17
```

Прыклад 6.6.
V. Праграма:

```
var a: array [1..20] of integer;
    n, min, k : integer;
begin
 write('Колькасць n =');
 readln(n); writeln('Лікі');
 for var i := 1 to n do
  read(a[i]);
 //пошук мінімальнага элемента
 min := a[1];
 for var i := 2 to n do
  if a[i] < min then
     min := a[i];
 //падлік колькасці
 k := 0;
 for var i := 1 to n do
  if a[i] = min then
     k := k + 1;
 writeln('Мінімальны ', min);
 writeln('Сустрэўся ',k,
         ' раз(-ы,  -оў)');
end.
```

6.2. Рашэнне задач з выкарыстаннем алгарытму пошуку максімальнага (мінімальнага) элементаў

Прыклад 6.5. У масіве захоўваецца інфармацыя пра вынікі спартсменаў, якія ўдзельнічаюць у лыжнай гонцы. Вызначыць вынік пераможцы і яго нумар.

I. Зыходныя даныя: масіў `a` — лікі, якія з'яўляюцца часам праходжання трасы, колькасць спартсменаў — `n`.

II. Вынік: `a[n_min]` — мінімальны час, `n_min` — нумар пераможцы.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. Для рашэння задачы выкарыстаем алгарытм пошуку мінімальнага элемента ў масіве і яго нумару (прыклад 6.4).
3. Вывад выніку.

IV. Апісанне пераменных: `a` — **array**`[1..20]` **of** `real`; `n`, `n_min` — `integer`.

Прыклад 6.6. Вызначыць, колькі разоў у лінейным масіве сустракаецца элемент, роўны мінімальнаму.

I. Зыходныя даныя: масіў `a`, колькасць лікаў `n`.

II. Вынік: `min` — мінімальны элемент, `k` — колькасць мінімальных.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. Пошук мінімальнага элемента.
3. Лінейны пошук элементаў, роўных мінімальнаму.
4. Вывад выніку.

IV. Апісанне пераменных: a — **array**[1..20] **of** integer; n, min, k — integer.

Прыклад 6.7. Зададзены масіў са слоў рознай даўжыні. Знайсці ў ім самае доўгае і самае кароткае слова.

I. Зыходныя даныя: масіў a, колькасць слоў n.

II. Вынік: min_s — самае кароткае слова, max_s — самае доўгае слова.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Пошук самага кароткага слова. Самае кароткае слова — слова, у якім мінімальная колькасць сімвалаў. Для яго пошуку можна выкарыстаць алгарытм пошуку мінімальнага элемента ў масіве. Аднак, калі параўноўваць самі элементы масіву, то параўнанне будзе адбывацца не па даўжыні[1]. Для параўнання радкоў па даўжыні трэба выкарыстоўваць функцыю вылічэння даўжыні радка length.

3. Для пошуку самага доўгага слова можна выкарыстоўваць алгарытм пошуку максімальнага элемента і параўноўваць элементы з выкарыстаннем функцыі вылічэння даўжыні радка length.

4. Вывад выніку.

IV. Апісанне пераменных: a — **array**[1..20] **of** string; n — integer; min_s, max_s: string;

Прыклад 6.6. *Працяг.*

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 5
Лікі
3 1 2 1 1
Мінімальны 1
Сустрэўся 3 раз(-ы)
```

Прыклад 6.7.

V. Праграма:

```
var a: array [1..20] of string;
    n: integer;
    min_s, max_s: string;
begin
 write('Колькасць n =');
 readln(n); writeln('Словы');
 for var i := 1 to n do
  readln(a[i]);
 //пошук  кароткага слова
 min_s := a[1];
 for var i := 2 to n do
  if length(a[i]) < length(min_s)
  then
    min_s := a[i];
 //пошук доўгага слова
 max_s := a[1];
 for var i := 2 to n do
  if length(a[i]) > length(max_s)
  then
    max_s := a[i];
 writeln('Кароткае - ',min_s);
 writeln('Доўгае - ',max_s);
end.
```

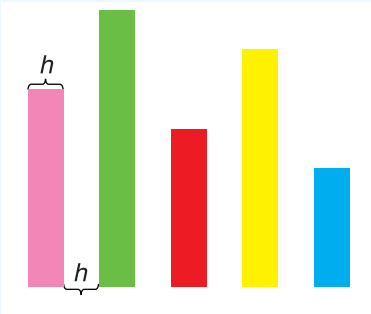
VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 5
Словы
Усе
дарогі
ідуць
у
Рым
Кароткае - у
```

[1] Параўнанне радкоў ажыццяўляецца **лексікаграфічна**: s1 < s2, калі для першага несупадаючага сімвала з нумарам i правільная няроўнасць s1[i] < s2[i] або ўсе сімвалы радкоў супадаюць, але s1 карацейшае за s2.

Прыклад 6.8.



V. Праграма:

```
uses graphABC;
var a: array[1..20] of integer;
  n, max, h, x, y1, y2: integer;
  m: real;
begin
  write('Колькасць n=');
  readln(n);
  writeln(n);
  writeln('Элементы масіву');
  for var i := 1 to n do
  begin
    read(a[i]);
    write(a[i],' ');
  end;
  max := a[1];
  for var i := 2 to n do
    if a[i] > max then
      max := a[i];
  h:= trunc(WindowWidth/(2*n+1));
  m:= WindowHeight/max;
  x:= h;
  for var i := 1 to n do
  begin
    SetBrushColor(clrandom);
    y1 := WindowHeight;
    y2 := y1-trunc(a[i]*m);
    Rectangle(x, y1, x+h, y2);
    x := x + 2*h;
  end;
end.
```

6.3. Пабудова гістаграмы (слупкаватай дыяграмы)

Прыклад 6.8. Дадзены аднамерны масіў з цэлых лікаў. Пабудаваць гістаграму па лікавых даных, якія захоўваюцца ў масіве.

I. Зыходныя даныя: масіў a, колькасць лікаў n.

II. Вынік: пабудаваная дыяграма.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Гістаграма складаецца з n прамавугольнікаў аднолькавай шырыні. Элементы масіву вызначаюць вышыню адпаведнага прамавугольніка. Максімальнае значэнне элементаў масіву (пераменная max) павінна па вышыні змясціцца ў акне (WindowHeight). Абазначым $m = \frac{\text{WindowHeigth}}{\text{max}}$ (маштабны каэфіцыент). Тады значэнню элемента масіву a[i] будзе адпавядаць цэлая частка ад велічыні a[i] * m.

3. Знаходзім максімальны элемент масіву.

4. У цыкле будуем прамавугольнікі. Усе прамавугольнікі маюць аднолькавую шырыню (h), адлегласць паміж імі можна вызначыць роўнай шырыні прамавугольніка. Тады шырыня прамавугольніка — цэлая частка ад дзялення шырыні акна на (2n + 1):

$$h = \frac{\text{WindowWidth}}{2n + 1}.$$

5. Пры вылічэнні вышыні прамавугольніка трэба ўлічыць тое, што вось y накіравана зверху ўніз.

6. Колер прамавугольніка будзем задаваць выпадковым чынам.

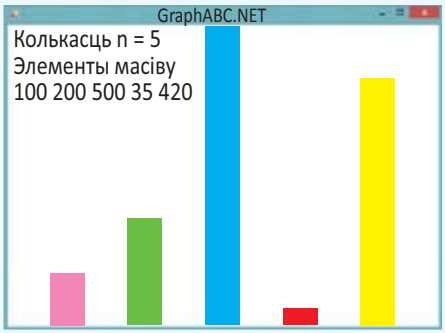
7. Месцазнаходжанне прамавугольніка вызначаецца пераменнай x. Пачатковае значэнне x = h. Новае значэнне атрымліваецца з папярэдняга павелічэннем на 2*h.

8. Вывад выніку.

IV. Апісанне пераменных: a — `array[1..20] of integer`; n, max, h, x, y1, y2 — `integer`; m: `real`.

Прыклад 6.8. *Працяг.*

VI. Тэсціраванне.



VII. Пабудуйце па гэтых даных дыяграму ў Excel і параўнайце.

1. Які элемент масіву з'яўляецца максімальным? Які — мінімальным?
2. Як знайсці максімальны элемент у масіве?
3. Як знайсці мінімальны элемент?
4. Якім чынам вызначыць нумар першага элемента, роўнага максімальнаму?
5. Як вызначыць нумар апошняга элемента, роўнага мінімальнаму?

Практыкаванні

1 Змяніце праграмы прыкладаў 6.1 і 6.2 так, каб знаходзіўся мінімальны элемент у масіве.

2 Для прыкладу 6.5 выканайце пералічаныя заданні.

1. Знайдзіце нумар спартсмена, які прыйшоў на фініш апошнім.
2. Вызначыце, ці быў пераможца адзіным або ёсць яшчэ лыжнік, які прайшоў трасу з такім жа вынікам (гл. прыклад 6.6).
3. Дабаўце яшчэ адзін масіў. Увядзіце ў яго прозвішчы спартсменаў. Рэалізуйце пункты 1—3 так, каб выводзілася прозвішча, а не нумар (гл. прыклад 5.10).

3 У масіве захоўваецца інфармацыя пра кошт аўтамабіляў. Вызначыце кошт самага дарагога аўтамабіля і яго нумар у масіве. Калі ёсць некалькі такіх аўтамабіляў, то выведзіце ўсе нумары.

4 У масіве захоўваецца інфармацыя пра сярэднядзённую тэмпературу снежня. Вызначыце, колькі ў снежні было дзён з самай нізкай і з самай высокай тэмпературай.

5 Зададзены масіў са слоў. Знайдзіце ў ім самае доўгае слова, якое заканчваецца літарай *а*.

6* Зададзены масіў са слоў. Знайдзіце ў ім самае кароткае слова, якое пачынаецца з вялікай літары.

7 Для прыкладу 6.8 выканайце пералічаныя заданні.

1. Змяніце праграму так, каб пры пабудове дыяграмы выкарыстоўвалася не ўся вышыня акна, а заставаліся палі зверху і знізу.
2. Змяніце праграму так, каб слупкі будаваліся без прамежкаў паміж імі.
3. Стварыце масіў колеравых канстант і выкарыстайце гэтыя колеры для зафарбоўвання слупкоў.
4. Пабудуйце лінейчастую дыяграму.

8 Па даных масіву пабудуйце дыяграму ў выглядзе ломанай лініі. Ці адпавядаюць ардынаты вяршынь ломаной значэнням масіву?

§ 7. Пераўтварэнне элементаў масіву

Прыклад 7.1.

V. Праграма:

```
var a: array[1..20] of integer;
    n: integer;
begin
 write('Колькасць n =');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 for var i := 1 to n do
 begin
  if a[i] > 0 then
     a[i] := a[i] * 2;
  if a[i] < 0 then
     a[i] := a[i] + 5;
 end;
 writeln('Пераўтвораны масіў');
 for var i := 1 to n do
  write(a[i], ' ');
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 5
Элементы масіву
3 -2 0 -1 5
Пераўтвораны масіў
6 3 0 4 10
```

VII. Аналіз вынікаў. Элементы 3 і 5 павялічаны ў 2 разы, элементы –2 і –1 павялічаны на 5, элемент 0 застаўся нязменным.

7.1. Асноўныя задачы

Сярод задач пераўтварэння элементаў масіву можна вылучыць задачы наступных тыпаў:

1. Змяненне элементаў масіву ў залежнасці ад умоў.
2. Абмен месцамі элементаў масіву.
3. Выдаленне элемента з масіву.
4. Устаўка элемента ў масіў.

Разгледзім кожную з задач.

7.2. Змяненне элементаў масіву ў залежнасці ад выканання некаторых умоў

Прыклад 7.1. Зададзены аднамерны масіў цэлых лікаў. Пераўтварыць яго элементы па наступным правіле: дадатныя элементы павялічыць у 2 разы, а адмоўныя — павялічыць на 5.

I. Зыходныя даныя: аднамерны масіў a, колькасць элементаў n.

II. Вынік: пераўтвораны масіў a.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.
2. У цыкле правяраем бягучы элемент. Калі ён дадатны, памнажаем яго на 2. Калі элемент

адмоўны, то прыбаўляем да яго 5. Памятаем, што адмаўленне ўмовы «элемент дадатны» — умова «элемент не дадатны», што мае на ўвазе магчымасць роўнасці элемента нулю. Таму патрэбныя два аператары галінавання для праверкі ўмовы задачы.

3. Вывад выніку.

IV. Апісанне пераменных: a — **array**[1..20] **of** integer; n — integer.

7.3. Абмен месцамі элементаў у масіве

Для абмену месцамі двух элементаў масіву можна выкарыстоўваць дадатковую пераменную, якую называюць буферам. Буферу прысвойваюць значэнне аднаго з элементаў масіву, гэтаму элементу прысвойваюць значэнне іншага элемента масіву, затым другому элементу прысвойваюць значэнне буфера:

```
buf := a[i];
a[i] := a[k];
a[k] := buf;
```

Прыклад 7.2. Зададзены аднамерны масіў цэлых лікаў. Памяняць месцамі максімальны і мінімальны элементы масіву (мінімальны і максімальны элементы сустракаюцца ў масіве толькі адзін раз).

I. Выходныя даныя: аднамерны масіў a, колькасць элементаў n.

II. Вынік: пераўтвораны масіў a.

III. Алгарытм рашэння задач.

1. Увядзём зыходныя даныя.

2. Знойдзем максімальны элемент масіву і яго індэкс (n_max).

Калі абмен элементаў ажыццяўляць наступным чынам:

```
a[i] := a[k]; a[k] := a[i];
```

то мы страцім значэнне элемента, які стаіць першапачаткова на месцы a[i], і атрымаем два элементы са значэннем, роўным a[k]. Для абмену элементаў можна выкарыстаць убудаваную функцыю swap: swap(a[i], a[k]).

Прыклад 7.2.
V. Праграма:

```
var a: array[1..20] of integer;
  n, n_min, n_max, buf: integer;
begin
 write('Колькасць n=');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 n_min := 1;
 n_max := 1;
 for var i := 1 to n do
 begin
  if a[i] > a[n_max] then
   n_max := i;
  if a[i] < a[n_min] then
   n_min := i;
 end;
 buf := a[n_min];
 a[n_min] := a[n_max];
 a[n_max] := buf;
 writeln('Пераўтвораны
        масіў');
 for var i := 1 to n do
  write(a[i], ' ');
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 5
Элементы масіву
2 1 3 5 4
Пераўтвораны масіў
2 5 3 1 4
```

Прыклад 7.3.

V. Праграма:

```
var a: array[1..20] of integer;
    n, d, j: integer;
procedure del_mas(k: integer);
begin
 for var i := k + 1 to n do
  a[i - 1] := a[i];
 n := n - 1;
end;
begin
 write('Колькасць n =');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 d := 0;
 j := 1;
 while j <= n do
 begin
  if a[j] mod 5 = 0 then
  begin
   del_mas(j);
   d := d + 1;
   j := j - 1;
  end;
  j := j + 1;
 end;
 writeln('Выдалілі ', d,
         'элемент(-ы, -аў');
 writeln('Пераўтвораны
          масіў');
 for var i := 1 to n do
  write(a[i], ' ');
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 7
Элементы масіву
5 3 15 35 10 4 30
Выдалілі 5 элемент(-ы,-аў)
Пераўтвораны масіў
3 4
```

VII. Аналіз вынікаў. Элементы 5, 15, 35, 10 і 30 кратныя 5, таму іх выдалілі з масіву. Элементы 3 і 4 не кратныя 5, таму яны засталіся ў масіве і зрушыліся адпаведна на 1-е і 2-е месцы.

3. Знойдзем мінімальны элемент масіву і яго індэкс (n_min).

4. Памяняем месцамі элементы, якія стаяць на месцах n_max і n_min.

5. Выведзем вынік.

IV. Апісанне пераменных: a — **array**[1..20] **of** integer; n, n_min, n_max, buf — integer.

7.4*. Выдаленне элемента з масіву

Для выдалення элемента масіву на месцы k трэба ссунуць на адну пазіцыю ўлева ўсе элементы, якія стаяць пасля яго. Колькасць элементаў пры гэтым памяншаем на 1.

```
for var i := k + 1 to n do
        a[i-1] := a[i];
n := n - 1;
```

Калі ў масіве трэба выдаліць не адзін, а некалькі элементаў, якія задавальняюць умову, то можна выкарыстоўваць дапаможны алгарытм у выглядзе адпаведнай працэдуры **procedure** del_mas(k: integer); Параметр k — нумар элемента, які выдаляецца.

Прыклад 7.3. Зададзены аднамерны масіў цэлых лікаў. Выдаліць з лінейнага масіву ўсе лікі, кратныя 5. Колькі лікаў выдалілі?

I. Зыходныя даныя: аднамерны масіў a, колькасць элементаў n.

II. Вынік: пераўтвораны масіў a і колькасць выдаленых лікаў d.

III. Алгарытм рашэння задачы.

1. Увядзём зыходныя даныя.

2. Будзем паслядоўна праглядаць элементы масіву. Калі знойдзем лік, кратны 5, то выдалім яго з масіву, выкарыстоўваючы працэдуру del_mas. Паколькі колькасць

элементаў, што выдаляюцца, загадзя не вядома, то выкарыстаем цыкл **while**.

3. Пры выдаленні элемента лічыльнік d будзем павялічваць на 1.

4. Выведзем вынік.

IV. Апісанне пераменных: a — **array**[1..20] **of** integer; n, d, j — integer.

7.5*. Устаўка элемента ў масіў

Для ўстаўкі элемента на месца k трэба вызваліць дадзенае месца ў масіве. Для гэтага ссунем на адну пазіцыю ўправа ўсе элементы масіву, якія стаяць пасля k – 1. Зрух пачынаем з апошняга элемента. Колькасць элементаў у масіве павялічыцца на 1.

Прыклад 7.4. Зададзены масіў цэлых лікаў. Уставіць лік x на k-е месца.

I. Зыходныя даныя: аднамерны масіў a, колькасць элементаў n, лік, які трэба ўставіць у масіў x, нумар пазіцыі ў масіве, на якую трэба ўставіць лік k.

II. Вынік: пераўтвораны масіў a.

III. Алгарытм рашэння задачы.

1. Увод зыходных даных.

2. Зрушваем усе элементы масіву, якія стаяць пасля k – 1, на адну пазіцыю ўправа.

3. Павялічым n — колькасць элементаў.

4. Устаўляем лік x на месца k.

5. Выводзім вынік.

IV. Апісанне пераменных: a — **array**[1..20] **of** integer; n, k, x — integer.

Прыклад 7.4.
V. Праграма:

```
var a: array[1..20] of integer;
    n, k, x: integer;
begin
 write('Колькасць n =');
 readln(n);
 writeln('Элементы масіву');
 for var i := 1 to n do
  read(a[i]);
 write('Лік x =');
 readln(x);
 write('Нумар пазіцыі k =');
 readln(k);
 //зрух элементаў управа на 1
 for var i := n downto k do
  a[i+1] := a[i];
 //устаўка x на месца k
 a[k] := x;
 n := n + 1;
 writeln('Пераўтвораны
    масіў');
 for var i := 1 to n do
  write(a[i], ' ');
end.
```

VI. Тэсціраванне.

Окно вывода

```
Колькасць n = 5
Элементы масіву
3 2 5 -3 7
Лік x = 6
Нумар пазіцыі k = 2
Пераўтвораны масіў
3 6 2 5 -3 7
```

Метады пашырэння аднамерных дынамічных масіваў дазваляюць пераўтвараць масівы з дапамогай убудаваных функцый ConvertAll, Replace, Transform і інш. (гл. даведачную сістэму PascalABC.Net).

1. Якія тыпы задач пераўтварэння масіваў вы можаце назваць?
2. Як можна памяняць месцамі два элементы ў масіве?
3. Як выдаліць элемент з масіву?
4. Як уставіць элемент у масіў?

Практыкаванні

1 Для задачы з прыкладу 7.1 выканайце пералічаныя заданні.

1. Запоўніце табліцу.

№	n	a	Пераўтвораны масіў
1	3	−2 −3 −5	
2	5	1 2 3 4 5	
3	10	1 −3 −2 0 4 0 2 −4 0 2	

2. Дабаўце ў табліцу свае значэнні n і a.

3. Ці можна замяніць каманды з п. 3.1 камандамі з п. 3.2?

3.1.
```
if a[i] > 0 then
  a[i] := a[i] * 2;
if a[i] < 0 then
  a[i] := a[i] + 5;
```

3.2.
```
if a[i] < 0 then
  a[i] := a[i] + 5;
if a[i] > 0 then
  a[i] := a[i] * 2;
```

4. У якіх выпадках праграма будзе даваць няправільны вынік?

2 Зададзены аднамерны масіў. Пераўтварыце яго элементы згодна з наступным правілам: ад усіх дадатных элементаў адняць элемент з нумарам k, да ўсіх адмоўных дадаць уведзены лік x. Нулявыя элементы пакіньце без змянення.

3 Зададзены аднамерны масіў з цотнай колькасці элементаў. Памяняйце месцамі яго «паловы».

4 У масіве запісаны прозвішчы і імёны навучэнцаў класа. З класа выбылі два навучэнцы. Вядомыя іх нумары. Выключыце даныя гэтых навучэнцаў з масіву.

5 Для задачы з прыкладу 7.4 выканайце пералічаныя заданні.

1. Запоўніце табліцу:

№	n	a	x	k	Пераўтвораны масіў
1	3	−2 −3 −5	0	2	
2	5	1 2 3 4 5	0	1	
3	10	1 −3 −2 0 4 0 2 −4 0 2	10	11	

2. Дабаўце ў табліцу свае значэнні n, a, x, k.

3. Які вынік выдасць праграма, калі ўвесці n = 5, a k = 120? Устаўце ў праграму праверку для ліку k ($1 < k \leqslant n + 1$).

4. Які вынік атрымаем, калі замяніць цыкл з п. 4.1 цыклам з п. 4.2?

4.1.
```
for var i := n downto k do
  a[i + 1] := a[i];
```

4.2.
```
for var i := k to n do
  a[i + 1] := a[i];
```

6 Перастаўце першы элемент масіву на апошняе месца, другі — на першае, трэці — на другое і г. д.

Глава 2
КАМП’ЮТАР ЯК УНІВЕРСАЛЬНАЕ ЎСТРОЙСТВА АПРАЦОЎКІ ІНФАРМАЦЫІ

§ 8. Апаратныя сродкі камп’ютара

8.1. Структурная схема камп’ютара

Правобразам першага камп’ютара прынята лічыць аналітычную машыну (прыклад 8.1), распрацаваную Чарльзам Бэбіджам у 1834 г. Бэбідж вызначыў структурныя элементы сучаснага камп’ютара: памяць, устройства для апрацоўкі даных (названае ім «млын») і ўстройствы для ўводу і выводу даных.

Па сутнасці, аналітычная машына з’яўляецца мадэллю разумовай дзейнасці чалавека. Працуючы з інфармацыяй, чалавек выконвае наступныя функцыі:

- прыём (г. зн. увод) інфармацыі;
- запамінанне (г. зн. захоўванне) інфармацыі;
- мысленне (г. зн. апрацоўка інфармацыі);
- перадача (г. зн. вывад) інфармацыі.

Камп’ютар з’яўляецца ўніверсальным устройствам для работы з данымі, таму ён павінен умець выконваць аналагічныя функцыі: увод, апрацоўку, захоўванне і вывад даных.

Пад **структурай камп’ютара** разумеюць мадэль, якая вызначае склад, парадак і прынцыпы ўзаемадзеяння элементаў камп’ютара.

Архітэктура камп’ютара — агульнае апісанне яго структуры і функцый.

Замест таго каб усё памятаць, чалавек пачаў рабіць пометкі: спачатку гэта былі наскальныя малюнкі, а са з’яўленнем пісьменства — кнігі. Такім чынам, даныя сталі захоўваць на знешніх носьбітах інфармацыі.

Для перадачы і атрымання інфармацыі выкарыстоўваліся розныя сігналы: запаленыя вогнішчы, гукі гонга і інш. У далейшым, з развіццём навукі, сігналы навучыліся перадаваць з дапамогай радыёхваль.

Апрацоўка інфармацыі патрабавала вылічальных дзеянняў, што прывяло да з’яўлення розных устройстваў для палягчэння лічэння: ад найпрасцейшага абака да сучасных камп’ютараў.

Прыклад 8.1. Аналітычная машына Бэбіджа.

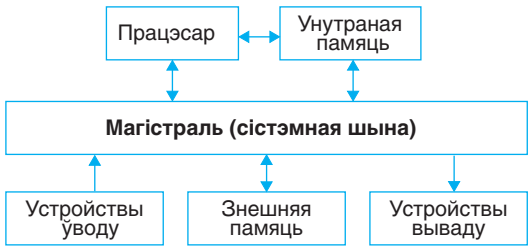
Архітэктура камп'ютара не ўключае ў сябе падрабязных апісанняў электронных схем. Гэтыя звесткі патрэбны канструктарам, спецыялістам па наладцы і рамонце камп'ютараў.

Сучасныя камп'ютары сабраны ў адпаведнасці з прынцыпам адкрытай архітэктуры. Гэты прынцып дазваляе збіраць камп'ютары, падбіраючы камплектуючыя ў залежнасці ад заяўленых крытэрыяў. Спецыфікацыі на стварэнне ўстройстваў распрацоўваюцца зацікаўленымі вытворцамі сумесна. Любы новы вузел (блок) можна замяніць, устанавіўшы ў тое ж месца ў камп'ютары, паколькі ён сумяшчаецца са старым.

Джон фон Нэйман (1903—1957) — матэматык, які зрабіў важны ўклад у інфарматыку і іншыя навукі. Найбольш вядомы як прабацька сучаснай архітэктуры камп'ютараў (архітэктура фон Нэймана).

Архітэктура фон Нэймана прадугледжвае адно ўстройства, праз якое праходзіць паток даных, і адно ўстройства кіравання, праз якое праходзіць паток каманд: SISD (*Single Instruction Single Data*) — «адзін паток каманд, адзін паток даных».

У паняцце «архітэктура камп'ютара» ўваходзяць: будова камп'ютара, фізічныя, арыфметычныя і лагічныя прынцыпы работы яго блокаў, склад і функцыі праграмнага забеспячэння. У аснову архітэктуры сучасных камп'ютараў пакладзены магістральна-модульны прынцып. Структурная схема камп'ютара мае наступны выгляд:



(Больш падрабязная структура камп'ютара паказана ў *Дадатку 2*, с. 114.)

У адпаведнасці з магістральна-модульным прынцыпам камп'ютар уяўляе сабой набор блокаў, якія ўзаемадзейнічаюць з агульным каналам для абмену данымі — сістэмнай шынай (магістраллю). Кожны блок выконвае спецыялізаваныя аперацыі.

Агульнасць архітэктуры розных камп'ютараў забяспечвае іх сумяшчальнасць з пункту гледжання карыстальніка.

Асноўныя прынцыпы архітэктуры камп'ютараў распрацаваны Д. фон Нэйманам у 1945 г. Прывядзём іх пералік:

1. Выкарыстанне двайковага кода.
2. Праграмнае кіраванне.
3. Захоўванне дадзеных праграм у памяці і аднолькавае кадзіраванне іх у двайковым кодзе.

4. Наяўнасць у ячэек памяці камп'ютара паслядоўна пранумараваных адрасоў.

5. Магчымасць умоўнага пераходу пры выкананні праграмы. Каманды выконваюцца паслядоўна, але пры неабходнасці можна рэалізаваць пераход да любой часткі кода.

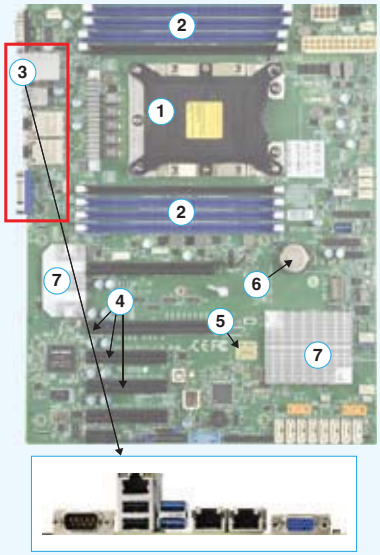
8.2. Сістэмная плата, сістэмная шына, працэсар

Сістэмная (мацярынская) плата з'яўляецца галоўнай платай у сістэмным блоку камп'ютара (прыклад 8.2). На ёй размяшчаюцца асноўныя кампаненты камп'ютарнай сістэмы (працэсар, аператыўная памяць, сістэмная шына і інш.). Мацярынская плата забяспечвае сувязь найважнейшых кампанентаў персанальнага камп'ютара паміж сабой.

На мацярынскай плаце маюцца спецыяльныя раздымы для ўстаноўкі ўнутраных устройстваў камп'ютара. Для падключэння кожнага ўстройства да мацярынскай платы распрацаваны розныя раздымы. У прыкладзе 8.2 нумарам «1» адзначаны **сокет** — раздым для размяшчэння працэсара. Нумар «2» паказвае на раздымы для ўстаноўкі аператыўнай памяці. Раздымы для падключэння знешніх устройстваў адзначаны нумарам «3».

Нумар «4» паказвае на слоты — раздымы для ўстаўкі карт расшырэння. **Карта расшырэння** — спецыяльная плата, якую ўстанаўліваюць у слот расшырэння мацярынскай платы з мэтай дабаўлення камп'ютару дадатковых функцый. Да плат расшырэння

Прыклад 8.2. Мацярынская плата камп'ютара.



Нумар «5» паказвае на мікрасхему, якая захоўвае BIOS — праграмнае забеспячэнне для пачатковай загрузкі камп'ютара. Нумарам «6» адзначана батарэйка, неабходная для падтрымання захаваных настроек мацярынскай платы. Таксама на мацярынскай плаце знаходзяцца чыпсеты (нумар «7») — мікрасхемы, якія дазваляюць працэсару абменьвацца інфармацыяй з памяццю і перыферыйнымі ўстройствамі.

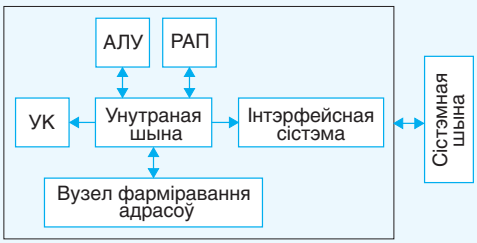
Сучасныя сістэмы ўключаюць два тыпы шын (архітэктура DIB — Dual independent bus, двайная незалежная шына):

- першасная шына (FSB, frontside bus), якая звязвае працэсар з аператыўнай памяццю і аператыўную памяць з перыферыйнымі ўстройствамі;
- другасная шына (BSB, backside bus) для сувязі з кэш-памяццю.

Выкарыстанне двайной незалежнай шыны павялічвае прадукцыйнасць працэсара, паколькі ў гэтым выпадку ён можа паралельна звяртацца да розных узроўняў памяці.

Прыклад 8.3. Працэсары.

Прыклад 8.4. Структурная схема працэсара (спрошчаная).



Унутраная шына мікрапрацэсара ажыццяўляе ўзаемасувязь паміж яго складальнікамі. Вузел фарміравання адрасу — блок, які адказвае за фарміраванне спісу адрасоў для выбару наступных каманд ці даных.

Канструкцыя мікрапрацэсара забяспечвае перадачу адрасоў, даных, каманд і кіруючых сігналаў. Па сістэмнай шыне ва ўстройства кіравання ўводзіцца код каманды. Затым сігнал дэшыфруецца і ствараецца паслядоўнасць мікракаманд, якую выконваюць блокі камп'ютара ці працэсар. Пры неабходнасці адначасова з гэтым фарміруецца адрас для загрузкі наступнай каманды ці даных.

Частата работы ўсіх сучасных працэсараў у некалькі разоў перавышае частату сістэмнай шыны, таму працэсар працуе так хутка, як яму гэта дазваляе сістэмная шына. Велічыню, на якую частата працэсара перавышае частату сістэмнай шыны, называюць множнікам.

належаць: відэакарта, гукавая карта, сеткавая карта і інш. Пры такой канструкцыі замена адных знешніх устройстваў на іншыя суправаджаецца простай заменай карты расшырэння.

Сістэмная шына выконвае ролю інфармацыйнай магістралі, якая злучае ўсе ўстройствы камп'ютара адзін з адным. Спрошчана сістэмную шыну можна паказаць як групу правадніковых і электрычных (токаправодных) ліній на сістэмнай плаце. Да сістэмнай шыны падключаны ўсе асноўныя блокі камп'ютара. Галоўнай функцыяй сістэмнай шыны з'яўляецца забеспячэнне ўзаемадзеяння паміж працэсарам і астатнімі кампанентамі камп'ютара. Па сістэмнай шыне ажыццяўляецца перадача даных, адрасоў памяці і кіруючых каманд. Частата шыны характарызуе прапускную здольнасць канала перадачы даных.

Цэнтральным устройствам камп'ютара з'яўляецца **працэсар** (прыклад 8.3). Ён непасрэдна выконвае аперацыі па апрацоўцы даных (арыфметычныя і лагічныя) і кіраванні вылічальным працэсам. Працэсар ажыццяўляе выбарку машынных каманд і даных з аператыўнай памяці, іх выкананне і запіс вынікаў назад у аператыўную памяць, кіруе знешнімі ўстройствамі.

Працэсар (мікрапрацэсар) уяўляе сабой мікрасхему, якая змяшчае ўстройства кіравання (УК), арыфметыка-лагічнае ўстройства (АЛУ) і рэгістры агульнага прызначэння (РАП). Структурная схема працэсара паказана ў прыкладзе 8.4.

Устройства кіравання выпрацоўвае кіруючыя сігналы для выканання зададзенай каманды мікрапрацэсарам і камп'ютарам у цэлым.

Арыфметыка-лагічнае ўстройства прызначана для выканання арыфметычных і лагічных аперацый апрацоўкі даных.

Рэгістры агульнага прызначэння — спецыяльныя ячэйкі звышхуткай памяці ўнутры працэсара, да якіх ён можа звяртацца напрамую; выкарыстоўваюцца пры выкананні арыфметычных аперацый.

Працэсары з'яўляюцца энергаёмістымі ўстройствамі і пры рабоце моцна награваюцца, таму на іх ставяць спецыяльныя сістэмы ахаладжэння (прыклад 8.5).

Асноўнымі характарыстыкамі працэсара з'яўляюцца:

- **тактавая частата** (паказвае хуткасць работы працэсара ў герцах (Гц), г. зн. вызначае колькасць рабочых аперацый у секунду);
- **разраднасць** (максімальная колькасць двайковых разрадаў, над якой адначасова можа выконвацца аперацыя перадачы і апрацоўкі даных);
- **памер кэш-памяці працэсара** (памер дадатковай высакаскораснай памяці, якая захоўвае копіі участкаў аператыўнай памяці, што выкарыстоўваюцца найбольш часта);
- **колькасць вылічальных ядзер** (кожнае ядро ўяўляе сабой частку працэсара, якая можа апрацоўваць асобны паток даных).

Прыклад 8.5. Сістэма ахаладжэння працэсара.

Сістэмы ахаладжэння таксама ўстанаўліваюць на іншыя кампаненты мацярынскай платы: відэакарту, чыпсет і інш.

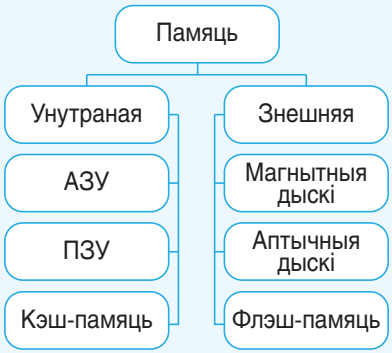
Першыя шмат'ядзерныя працэсары ўяўлялі сабой самыя простыя схемы: два працэсарныя ядры, змешчаныя на адным крышталі без падзелу якіх-небудзь рэсурсаў, акрамя шыны памяці.

Сёння існуюць два часта ўжываемыя тэрміны для працэсараў, якія маюць некалькі ядзер: мультыядзерны, або шмат'ядзерны (multi-core), і шматпрацэсарны (many-core). Пад шматядзернасцю працэсара разумеюць, што некалькі ядзер з'яўляюцца інтэгравanымі на адну інтэгральную схему. Тэрмінам *шматпрацэсарны* абазначаюць камп'ютары, якія маюць некалькі фізічна падзеленых працэсараў, што кіруюцца адным экземплярам аперацыйнай сістэмы (АС).

Вылічальныя ядры шмат'ядзернага працэсара сумесна выкарыстоўваюць кэш трэцяга ці другога ўзроўню.

У жніўні 2019 г. кампанія Cerebras паказала самы вялікі ў свеце шмат'ядзерны суперпрацэсар Cerebras Wafer Scale Engine. Ён мае больш за 1,2 трлн транзістараў і 400 000 ядзер і займае амаль усю плошчу паўправадніковай пласціны дыяметрам 300 мм.

Прыклад 8.6. Віды камп'ютарнай памяці:



Прыклад 8.7. Аператыўная памяць:

Аператыўная памяць з'яўляецца хуткадзейнай і дазваляе звяртацца да кожнай ячэйкі памяці асобна (прамы доступ да ячэйкі па адрасе).

Прыклад 8.8. Пастаянная памяць:

Найважнейшая мікрасхема ПЗУ — модуль BIOS (ад англ. basic input/output system — базавая сістэма ўводу/вываду).

На мацярынскай плаце таксама ўстаноўлена CMOS (паўпастаянная памяць) — памяць для захоўвання параметраў канфігурацыі камп'ютара і бягучага часу.

8.3. Віды і прызначэнне памяці

Камп'ютарная памяць служыць для захоўвання даных і бывае некалькіх тыпаў. Кожны тып памяці прызначаны для выканання розных задач.

Адрозніваюць унутраную і знешнюю памяць (прыклад 8.6). Да **ўнутранай памяці** залічваюць:

- аператыўную памяць (аператыўнае запамінальнае ўстройства — АЗУ, англ. *random access memory — RAM*);
- пастаянную памяць (пастаяннае запамінальнае ўстройства — ПЗУ, англ. *read only memory — ROM*);
- кэш-памяць.

Знешнюю памяць падзяляюць паводле фізічных прынцыпаў запісу даных: магнітныя носьбіты, аптычныя носьбіты, флэш-памяць.

Аператыўная памяць служыць для захоўвання праграм і даных, з якімі працэсар працуе ў дадзены момант (прыклад 8.7). Сучасныя тыпы аператыўнай памяці не могуць захоўваць даныя пасля выключэння сілкавання камп'ютара — памяць энергазалежная. Аб'ём аператыўнай памяці сучасных камп'ютараў складае 4—64 Гбайт.

Пастаянная памяць захоўвае праграмы аўтаматычнага тэсціравання ўстройстваў і загрузкі АС у аператыўную памяць (прыклад 8.8). ПЗУ з'яўляецца энерганезалежнай памяццю, паколькі захоўвае інфармацыю пасля адключэння сілкавання камп'ютара. У большасць мікрасхем ПЗУ немагчыма ўнесці змяненні. Мае невялікі аб'ём — ад 384Кб да 8 Мб.

Кэш-памяць — хуткадзейная памяць, якая дазваляе павялічыць хут-

касць выканання аперацый. Служыць буферам паміж апаратыўнай памяццю і мікрапрацэсарам.

Знешняя памяць прызначана для працяглага захоўвання інфармацыі, з'яўляецца энерганезалежнай, мае вялікія памеры (да некалькіх Тэрабайт).

Да магнітных носьбітаў належыць **вінчэстар** (накапляльнік на цвёрдых магнітных дысках — НЦМД, англ. *hard disk drive — HDD*). Ён уяўляе сабой сукупнасць з некалькіх дыскаў (пласцін) з нанесенымі магнітнымі пластамі (прыклад 8.9). Дыскі размяшчаюцца на адной восі электрарухавіка і знаходзяцца ў спецыяльным металічным корпусе.

Шырокае распаўсюджванне атрымалі знешнія вінчэстары, якія выкарыстоўваюць для падключэння да камп'ютара раздым USB. Шмат якія з іх аб'ядноўваюць традыцыйны цвёрды дыск з модулем флэш-памяці, што дазваляе павялічыць хуткасць яго работы.

Даныя на аптычныя носьбіты запісваюцца з дапамогай лазера. Найбольш вядомыя тыпы аптычных дыскаў — CD, DVD і Blu-ray (прыклад 8.10).

Флэш-памяць — паўправадніковая памяць, пабудаваная на аснове інтэгральных мікрасхем (прыклад 8.11). Флэш-памяць кампактная і даўгавечная, мае высокую хуткадзейнасць. Яе выкарыстоўваюць у лічбавых фота- і відэакамерах, мабільных тэлефонах і г. д.

Шмат у якіх сучасных камп'ютарах устанаўліваюць цвердацельныя накапляльнікі SSD (Solid State Drive), якія зарэкамендавалі сябе як больш надзейныя і хуткія альтэрнатывы HDD. Унутраная будова SSD уяўляе

Кэш-памяць падзяляецца на тры ўзроўні: L1, L2, L3. Кожны з узроўняў адрозніваецца па памеры памяці, хуткасці, выкананых задачах. L1 (устанаўліваецца на працэсары) — самы маленькі (да 128 Кбайт) і хуткі, L2 (можа змяшчацца на тым жа крышталі, што і працэсар, ці быць асобнай мікрасхемай) — сярэдні (ад 256 кбайт да 12 Мбайт), L3 — самы вялікі (0—16 Мбайт) і марудны, устанаўліваецца на серверах. Да кожнага ўзроўню працэсар звяртаецца па чарзе (ад меншага да большага), пакуль не выявіць у адным з іх патрэбныя даныя. Калі нічога не знойдзена, то працэсар звяртаецца да апаратыўнай памяці.

Прыклад 8.9. Вінчэстар:

Прыклад 8.10. Аптычныя дыскі:

Прыклад 8.11. Флэш-памяць:

Тэхналогію флэш-памяці выкарыстоўваюць наступныя віды ўстройстваў:

- compactFlash (выкарыстоўваецца ў лічбавых фотаапаратах);
- microSD/miniSD (флэш-карта, якая выкарыстоўваецца ў мабільных тэлефонах);
- знешнія накапляльнікі (флэшкі, падключаюцца да камп'ютарнай тэхнікі з дапамогай USB-раздыму).

Прыклад 8.12. Цвердацельны SSD.

Прыклад 8.13. Схема перадачы даных ад знешніх устройстваў да працэсара мае наступны выгляд:

сабой набор мікрасхем флэш-памяці, змешчаных на адной плаце (прыклад 8.12).

Знешняя памяць прызначана для працяглага захоўвання праграм і даных, і цэласнасць яе змесціва не залежыць ад таго, уключаны ці выключаны камп'ютар. У адрозненне ад аператыўнай памяці яна не мае прамой сувязі з працэсарам. Даныя ад знешніх устройстваў (ЗУ) да працэсара і назад перадаюцца праз аператыўную памяць (прыклад 8.13).

1. Што разумеюць пад структурай і архітэктурай камп'ютара?
2. Якія прынцыпы архітэктуры камп'ютара сфармуляваў Д. фон Нэйман?
3. Якія ўстройствы ўваходзяць у склад працэсара?
4. Якое прызначэнне сістэмнай шыны?
5. На якія віды падзяляецца камп'ютарная памяць?

Практыкаванні

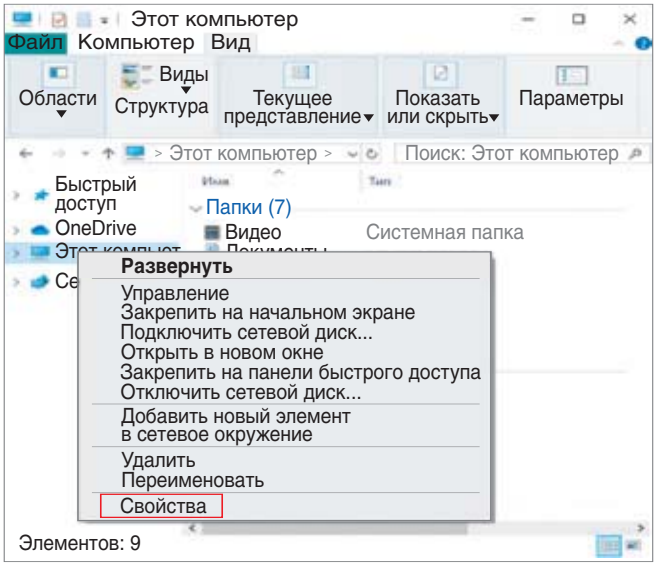
1 Падрыхтуйце ў рэжыме сумеснага доступу прэзентацыю на адну з тэм.

1. Гісторыя носьбітаў інфармацыі.
2. Віды камп'ютарнай памяці.
3. Пакаленні ЭВМ.

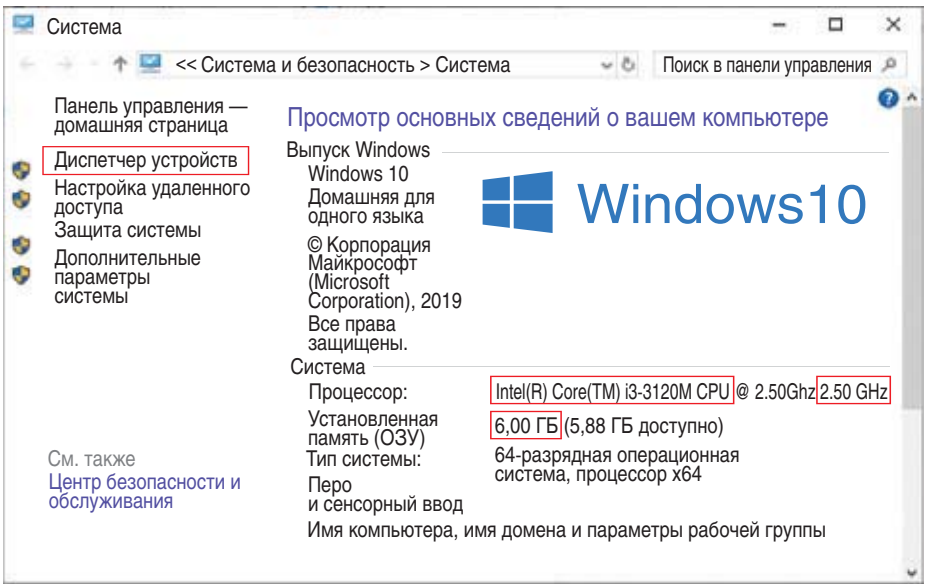
2 Запоўніце табліцу. (Працаваць з табліцай рэкамендуецца, выкарыстоўваючы воблачныя тэхналогіі.)

	Школьны камп'ютар	Дамашні камп'ютар
Працэсар		
Частата працэсара		
Аб'ём аператыўнай памяці		
Ёмістасць дыска C:		
Свабодна на дыску C:		
Іншыя ўстройствы знешняй памяці		

Даныя для табліцы можна атрымаць, адкрыўшы ўласцівасці камп'ютара ў праграме **Проводник**:

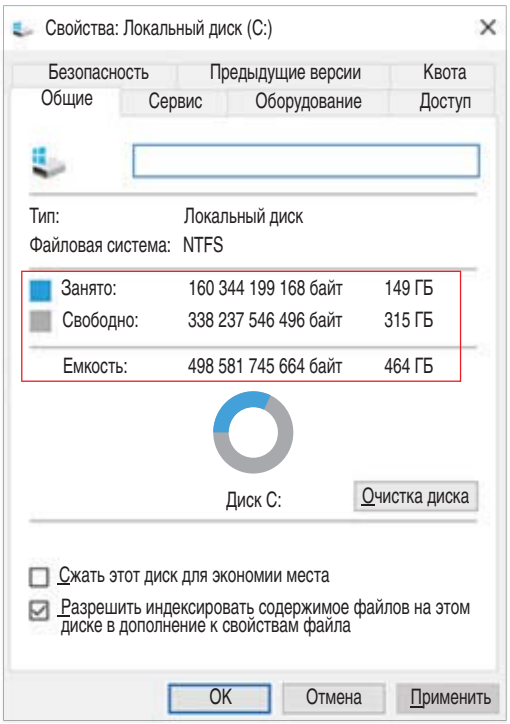


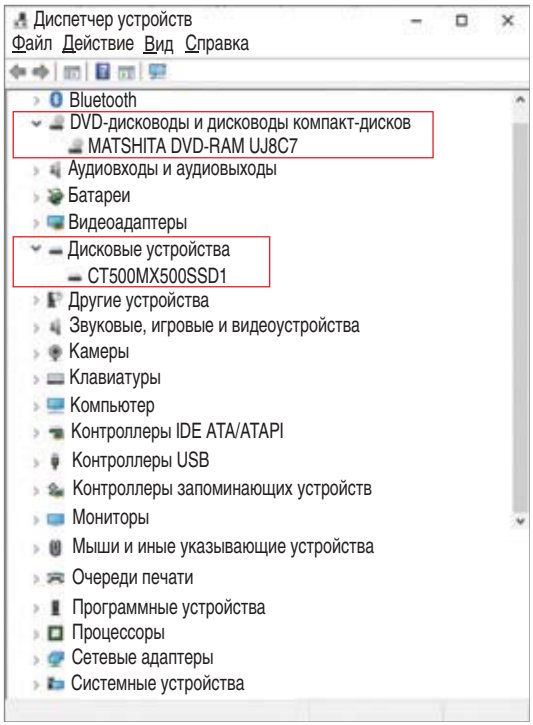
Вызначыце тып і характарыстыкі працэсара, а таксама памер аператыўнай памяці, установленай на школьным і вашым дамашнім (пры яго наяўнасці) камп'ютарах.



Вызначыце аб'ём дыска C: і колькасць свабоднай памяці на ім, выкарыстоўваючы ўласцівасці дыска (каманда **Свойства** ў кантэкставым меню дыска C:).

*Адкрыйце **Диспетчер устройств** і вызначыце, якія яшчэ ўстройствы знешняй памяці падключаны да камп'ютара.

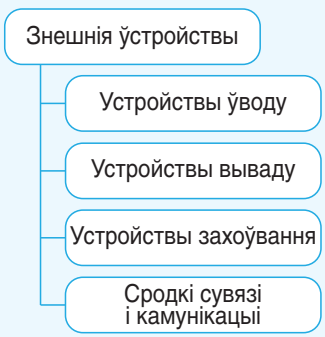




3* Адкрыйце сайт (які назаве настаўнік) з віртуальным трэнажорам па зборцы ПК і выканайце заданні.

§ 9. Знешнія ўстройствы

Прыклад 9.1. Знешнія ўстройствы:



9.1. Класіфікацыя знешніх устройстваў

Знешнія (перыферыйныя) устройствы забяспечваюць узаемадзеянне камп'ютара з карыстальнікам, іншымі камп'ютарамі або тэхнічнымі ўстройствамі. Яны падключаюцца да камп'ютара праз спецыяльныя раздымы — парты ўводу-выводу. Большасць сучасных знешніх устройстваў падключаецца да порта USB (Universal

Serial Bus — універсальная паслядоўная шына).

Знешнія ўстройствы па сваім прызначэнні можна падзяліць на: устройствы ўводу, устройствы вываду, устройствы захоўвання інфармацыі, а таксама сродкі сувязі і камунікацыі (прыклад 9.1).

Да **ўстройстваў уводу** інфармацыі належаць:

- клавіятура;
- устройствы ўказання, або графічныя маніпулятары: джойсцік (прыстасаванне ў выглядзе рычага — дзяржальна, штурвала, — якое дазваляе кіраваць віртуальным аб'ектам у двух- ці трохмернай прасторы); светлавое пяро (маніпулятар, які дазваляе ўводзіць інфармацыю шляхам дакранання устройства да экрана); мыш; трэкбол («мыш наадварот»: для работы неабходна круціць шар, замацаваны ў нерухомым корпусе);
- графічны планшэт, або дыгітайзер (складаецца з пяра і планшэта, адчувальнага да націскання ці блізкасці пяра; прызначаны для ўводу ў камп'ютар інфармацыі, створанай «ад рукі»);
- сканер;
- мікрафон.

(Разгледзьце прыклад 9.2.)

Да **ўстройстваў вываду** інфармацыі належаць:

- манітор;
- прынтар (лазерныя і струменныя прынтары дазваляюць выводзіць інфармацыю на паперу, 3D-прынтары дазваляюць ствараць аб'ёмныя аб'екты);

Прыклад 9.2. Устройствы ўводу:

Клавіятура

Устройствы ўказання:

Джойсцік

Светлавое пяро

Мыш

Трэкбол

Графічны планшэт

Сканер

Мікрафон

Прыклад 9.3. Устройствы вываду:

Манітор

Прынтар

3D-прынтар

Плотар

Калонкі

Навушнікі

Прыклад 9.4. Платы расшырэння.

Гукавая карта

Відэакарта

• плотар (устройства для аўтаматычнага вычэрчвання малюнкаў, схем, чарцяжоў, карт; рэжучы плотар дазваляе ажыццяўляць выразку лякалаў з кардону, скуры, пластыку і інш.);

• калонкі, навушнікі.

(Разгледзьце прыклад 9.3.)

Шмат якія з устройстваў камбінуюць у сабе ўстройствы ўводу і вываду: шматфункцыянальнае ўстройства (ШФУ) змяшчае ў сабе сканер і прынтар; гарнітура аб'ядноўвае мікрафон і навушнікі. Сэнсарныя маніторы не толькі адлюстроўваюць інфармацыю, але і дазваляюць яе ўводзіць.

Устройствы захоўвання — устройствы знешняй памяці, якія былі разгледжаны ў папярэднім параграфе.

Для работы шмат якіх устройстваў неабходны карты (платы) расшырэння (прыклад 9.4), якія змяшчаюць адаптары. **Адаптар** неабходны для пераўтварэння сігналу, які паступае ад устройства, у двайковы код і назад. Некаторыя адаптары ўбудаваны непасрэдна на мацярынскую плату.

Да **сродкаў сувязі і камунікацыі** залічваюць устройствы, якія дазваляюць арганізаваць перадачу даных па камп'ютарнай сетцы:

• сеткавы адаптар, або сеткавая карта (прызначаецца для злучэння камп'ютараў у лакальную сетку па тэхналогіі Ethernet — правадное злучэнне);

• мадэм (устройства перадачы даных паміж камп'ютарамі па тэлефоннай і іншым лініям сувязі);

• маршрутызатар, або роўтар (устройства, неабходнае для перанакіравання пакетаў даных у адной ці не-

калькіх падсетках; маршрутызатары дазволяць забяспечыць як правадны, так і бесправадны доступ у Інтэрнэт).

(Разгледзьце прыклад 9.5.)

9.2. Апаратнае забеспячэнне для падключэння да сеткі Інтэрнэт

Колькасць карыстальнікаў Інтэрнэту ў сучасным свеце імкліва расце. Сёння практычна кожны чалавек можа падключыцца да Інтэрнэту.

Інтэрнэт — складаная сістэма камп'ютарных сетак. Перадача даных у камп'ютарных сетках патрабуе ўзгодненай работы вялікай колькасці разнастайных устройстваў. Для зладжанага ўзаемадзеяння работы сеткавых устройстваў Міжнароднай арганізацыяй стандартаў (International Standard Organization — ISO) была распрацавана сеткавая мадэль OSI (Open System Interconnection). Гэта мадэль апісвае правілы і спосабы перадачы даных у розных сеткавых асяроддзях пры арганізацыі сеанса сувязі. Асноўнымі элементамі мадэлі з'яўляюцца ўзроўні, прыкладныя працэсы і фізічныя сродкі злучэння (прыклад 9.6). Мадэль уключае ў сябе сем узроўняў. Кожнаму з іх адводзіцца канкрэтная роля, а агульная задача перадачы даных разбіваецца на асобныя падзадачы. Асноўным з пункту гледжання карыстальніка з'яўляецца прыкладны ўзровень. Гэты ўзровень забяспечвае выкананне прыкладных задач карыстальнікаў. На ім рэалізуюцца такія сэрвісы, як аддаленая перадача даных, электронная пошта і работа вэб-браўзераў.

Прыклад 9.5. Устройствы камунікацыі.

Сеткавая карта

4G-мадэм для падключэння па каналах сотавай сувязі

Маршрутызатар з убудаваным мадэмам

Сучасны мабільны тэлефон можа выступаць у якасці роўтара і забяспечыць доступ у Інтэрнэт для іншых устройстваў.

Прыклад 9.6. Узроўні сеткавай мадэлі OSI:

Узровень пратакола	Адзінка вымярэння даных (pdu — protocol data units)
Фізічны	Біты
Канальны	Фрэймы
Сеткавы	Пакеты
Транспартны	Блокі
Сеансавы	Даныя
Прадстаўнічы	Даныя
Прыкладны	Даныя

Узроўні сеткавай мадэлі OSI

1. **Фізічны ўзровень** звязаны з работай апаратных сродкаў і вызначае фізічныя аспекты перадачы інфармацыі па лініях сувязі (узровень напружання і частаты, прыроду перадавальнага асяроддзя, спосаб перадачы двайковых даных па фізічным носьбіце). Для падтрымкі фізічнага ўзроўню ў камп'ютары ўстанаўліваюць сеткавы адаптар.

2. **Канальны ўзровень** адказвае за перадачу даных па фізічным узроўні з праверкай магчымасці перадачы даных, рэалізуе механізм выяўлення і карэкцыі памылак паміж вузламі сеткі. За пратаколы канальнага ўзроўню адказваюць сеткавыя адаптары і іх драйверы.

3. **Сеткавы ўзровень** адказвае за дастаўку інфармацыі ад вузла-адпраўшчыка да вузла-атрымальніка, забяспечвае маршрутызацыю пакетаў даных.

4. **Транспартны ўзровень** забяспечвае дастаўку інфармацыі вышэйлеглым узроўням з неабходнай ступенню надзейнасці. Можа выяўляць і выпраўляць памылкі перадачы, такія як скажэнне, страта ці дастаўка пакетаў у няправільным парадку.

5. **Сеансавы ўзровень** выконвае задачу арганізацыі сеансаў: усталяванне, падтрыманне і завяршэнне злучэння паміж дадаткамі ўдзельнікаў злучэння.

6. **Прадстаўнічы ўзровень** адказвае за форму падання даных, іх шыфроўку/дэшыфроўку, забяспечвае сцісканне/распакоўванне і перакадзіроўку даных з аднаго фармату ў іншы.

7. **Прыкладны ўзровень** забяспечвае ўзаемадзеянне карыстальніцкіх дадаткаў з сеткай.

Сёння існуюць разнастайныя спосабы падключэння да Інтэрнэту. Асноўныя адрозненні: прынцып работы, хуткасць перадачы даных, надзейнасць, складанасць настройкі абсталявання, кошт.

Паводле колькасці трафіку выкарыстанне Інтэрнэту можна падзяліць на дзве групы: прагляд старонак (малая колькасць трафіку) і спампоўванне вялікіх файлаў — фільмаў, музыкі і г. д. (патрабуе вялікай колькасці трафіку). У першым выпадку дастаткова скорасці звычайнага мадэмнага злучэння. Аднак такая скорасць абцяжарвае спампоўванне файлаў, таму для паўнавартаснага выкарыстання магчымасцей Інтэрнэту патрабуецца высакаскорасны доступ.

Існуе два віды тэхналогій выхаду ў Інтэрнэт:

1) правадная;

2) бесправадная.

Ад тэхналогіі падключэння да Інтэрнэту залежыць тып мадэма, які трэба выкарыстоўваць (прыклад 9.7).

Перадача даных пры правадной тэхналогіі ажыццяўляецца па спецыяльным кабелі (оптавалакно ці вітая пара), які з аднаго боку падключаны да абсталявання правайдара, а з іншага — у сеткавую карту камп'ютара.

У залежнасці ад тыпу абсталявання правайдара выкарыстоўваюцца наступныя спосабы праваднога падключэння да Інтэрнэту:

1. Мадэмныя злучэнні (ADSL). Дадзеная тэхналогія ператварае аналагавыя сігналы, якія перадаюцца па стандартнай тэлефоннай лініі, у лічбавыя

сігналы (пакеты даных). Пры гэтым у час працы можна рабіць званкі.

2. Злучэнне па вылучанай лініі. Пры гэтым карыстальнік атрымлівае пастаянны выхад у Інтэрнэт праз асобную свабодную тэлефонную лінію, якая гарантуе высокую якасць злучэння і перадачу даных на высокай скорасці.

3. Падключэнне праз тэлевізійны кабель. Гэты спосаб магчымы толькі ў выпадку наяўнасці кабельнага тэлебачання.

Бесправадныя тэхналогіі служаць для перадачы даных паміж двума і больш пунктамі на адлегласці, не патрабуючы правадной сувязі. Для перадачы інфармацыі могуць выкарыстоўвацца радыёхвалі, а таксама інфрачырвонае, аптычнае ці лазарнае выпраменьванне. Найбольш істотнымі характарыстыкамі бесправадных тэхналогій перадачы даных з'яўляюцца максімальная скорасць перадачы і максімальная адлегласць, на якую можна перадаваць інфармацыю (прыклад 9.8).

Бесправадныя маршрутызатары дазваляюць выкарыстоўваць Інтэрнэт, знаходзячыся ў любым месцы ў межах зоны доступу.

Яшчэ нядаўна падключэнне да Інтэрнэту праз спадарожнікавую сувязь было практычна недаступна для звычайных карыстальнікаў з прычыны высокага кошту.

Сучасная спадарожнікавая сувязь па скорасці, надзейнасці і бяспецы не саступае традыцыйнай правадной. Істотнай яе перавагай з'яўляецца магчымасць ужывання ў аддаленных і цяжкадаступных месцах, дзе немагчыма

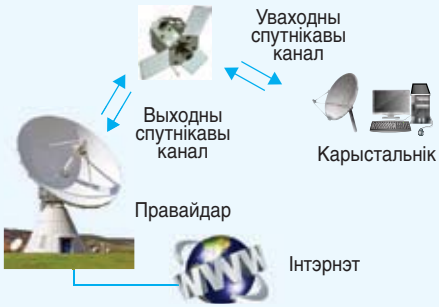
Прыклад 9.7. Класіфікацыя мадэмаў паводле віду злучэння:

- для лічбавых камутацыйных тэлефонных ліній;
- для арганізацыі вылучанай лініі ў тэлефоннай сетцы;
- кабельныя;
- радыёмадэмы;
- спадарожнікавыя.

Прыклад 9.8. Класіфікацыя бесправадных сетак паводле далёкасці дзеяння:

- Бесправадныя персанальныя сеткі (Wireless Personal Area Networks — WPAN) заснаваны на тэхналогіі Bluetooth, якая дазваляе ўстройствам падтрымліваць сувязь са скорасцю да 24 Мбіт/с, калі яны знаходзяцца ў радыусе да 10 м адно ад аднаго.
- Бесправадныя лакальныя сеткі (Wireless Local Area Networks — WLAN) выкарыстоўваюць тэхналогію Wi-Fi. Тэхналогія хутка развіваецца. У канцы 2018 г. быў паказаны стандарт пакалення Wi-Fi 6, які дазваляе перадаваць даныя са скорасцю да 11 Гбіт/с у радыусе 300 м.
- Бесправадныя сеткі маштабу горада (Wireless Metropolitan Area Networks — WMAN) выкарыстоўваюць тэхналогію WiMAX з ахопам тэрыторыі да 150 км і скорасцю перадачы даных да 1 Гбіт/с.
- Бесправадныя глабальныя сеткі (Wireless Wide Area Network — WWAN) засноўваюцца на розных тэхналогіях, такіх як GPRS, EDGE, HSPA, LTE і інш., якія з'яўляюцца надбудовамі над тэхналогіяй мабільнай сувязі. Яны дазваляюць карыстальніку сеткі сотавай сувязі рабіць абмен данымі з іншымі ўстройствамі ў сетцы, а таксама са знешнімі сеткамі, у тым ліку Інтэрнэт. У якасці мадэма для падключэння да Інтэрнэту можа выкарыстоўвацца мабільны тэлефон. Камп'ютар можа падключацца да тэлефона з дапамогай USB-кабеля ці бесправадным спосабам.

Прыклад 9.9. Схема падключэння да Інтэрнэту з выкарыстаннем спадарожнікавай сувязі:



Двайковы код выкарыстоўваецца для кадзіравання даных яшчэ і таму, што ўстройствам значна прасцей і хутчэй выконваць арыфметыка-лагічныя аперацыі ў двайковай сістэме злічэння, чым у дзесятковай.

У 1959 г. вучоныя з Маскоўскага дзяржаўнага ўніверсітэта пад кіраўніцтвам Мікалая Брусенцава распрацавалі першую і адзіную ЭВМ на аснове трайковай логікі. Называлася яна «Сетунь». Іншых камп'ютараў на аснове трайковага кода няма і не было.

Ідэю выкарыстоўваць для вылічэнняў трайковую сістэму выказаў яшчэ ў XIII ст. італьянскі матэматык Фібаначы. Ён сфармуляваў і вырашыў задачу пра гіры: калі можна класці гіры толькі на адну чашу шаляў, то зручней, хутчэй і эканамічней рабіць падлікі ў двайковай сістэме, а калі можна класці гіры на абедзве чашы, то мэтазгодней звярнуцца да трайковай сістэмы.

ці занадта дорага пракладаць інтэрнэт-кабель. Спадарожнікавай сувязю шырока карыстаюцца дзяржаўныя службы, геолагаразведачныя і нафтавыя кампаніі, тэлекамунікацыйныя фірмы і іншыя буйныя арганізацыі. У шэрагу выпадкаў яна з'яўляецца адзіным варыянтам інтэрнэт-камунікацый (прыклад 9.9).

9.3. Прынцыпы работы апаратных сродкаў камп'ютара

Сучасны камп'ютар выкарыстоўвае электрычныя сігналы, г. зн. ток ці напружанне, значэнні якіх змяняюцца па законе, што адлюстроўвае перадаваемае паведамленне. З дапамогай сігналаў кадзіруюцца даныя, якія перадаюцца і апрацоўваюцца. Электрычныя сігналы можна выкарыстоўваць для кадзіравання як двайковы код: «1» — ёсць ток (ток большы за парогавую велічыню); «0» — няма току (ток меншы за парогавую велічыню).

Чым менш значэнняў існуе ў сістэме, тым прасцей вырабіць асобныя канструктыўныя элементы, якія апераруюць гэтымі значэннямі. Найбольш надзейным і танным з'яўляецца ўстройства, кожны разрад якога можа прымаць два станы: ёсць ток/няма току, высокае напружанне/нізкае напружанне, намагнічана/не намагнічана і г. д.

? **1.** Якія ўстройствы камп'ютара залічваюць да знешніх?
2. На якія групы можна падзяліць знешнія ўстройствы ў залежнасці ад іх прызначэння?
3. Якія існуюць спосабы падключэння да Інтэрнэту?
4. Якое абсталяванне можа выкарыстоўвацца для праваднога падключэння да Інтэрнэту?
5. Якія тэхналогіі выкарыстоўваюцца для бесправаднога падключэння да Інтэрнэту?

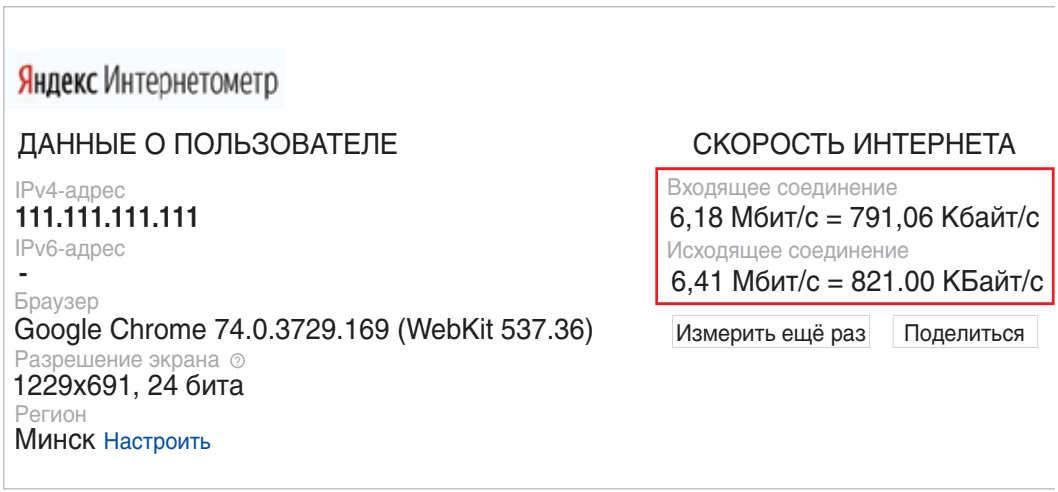
Практыкаванні

1 Падрыхтуйце прэзентацыю на адну з пералічанных тэм у рэжыме сумеснага доступу.

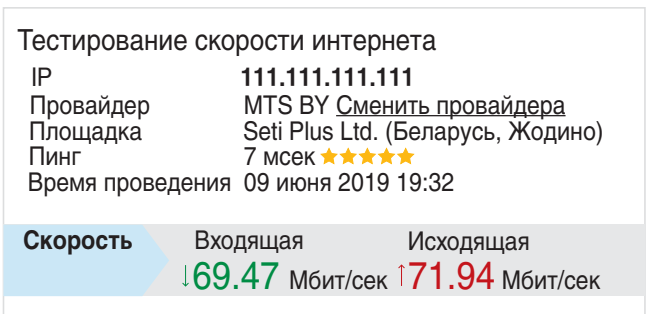
1. Прынтары.
2. Спосабы падключэння да Інтэрнэту.
3. Перавагі і недахопы бесправаднога падключэння да Інтэрнэту.
4. Інтэрнэт праз спутнік.
5. Прынцып работы ЭВМ «Сетунь».

2 Вызначыце скорасць падключэння вашага ўстройства да Інтэрнэту, выкарыстоўваючы магчымасці пералічаных сайтаў (або іншых аналагічных).

1. https://yandex.by/internet/



2. https://2ip.ru/speed/



§ 10. Праграмнае забеспячэнне камп'ютара

Упершыню ідэя пра паасобны разгляд каманд і даных была выказана Чарльзам Бэбіджам у XIX ст. Пазней, у XX ст., яна была развіта ў прынцыпах Джона фон Нэймана. Гэтыя прынцыпы ўлічваюцца і пры распрацоўцы архітэктур сучасных камп'ютараў, і пры распрацоўцы камп'ютарных праграм.

Сумеснае выкарыстанне шыны для памяці праграм і памяці даных прыводзіць да «вузкага месца архітэктуры фон Нэймана». З прычыны таго, што памяць праграм і памяць даных не могуць быць даступныя ў адзін і той жа час, прапускная здольнасць канала «працэсар — памяць» істотна абмяжоўвае хуткасць работы камп'ютара.

Вучоныя з ЗША і Італіі ў 2015 г. заявілі пра стварэнне прататыпа мемпрацэсара (mem — ад англ. *memory*) з архітэктурай, якая адрозніваецца ад архітэктуры фон Нэймана. Мем-працэсар рэалізуе адначасовае вылічэнне і захоўванне атрыманых даных у адным месцы шляхам узаемадзеяння ячэек памяці.

Прыклад 10.1. Работа камп'ютара кіруецца праграмай, якая складаецца з набору каманд. Каманды запісваюцца ў памяць камп'ютара і выконваюцца паслядоўна, адна за адной. Паслядоўнасць парушаецца толькі ў тым выпадку, калі выконваецца каманда ўмоўнага ці безумоўнага пераходу. У камандзе пераходу непасрэдна паказваецца адрас наступнай каманды. Працэс вылічэнняў працягваецца датуль, пакуль не будзе выканана каманда, якая прадпісвае заканчэнне вылічэнняў.

10.1. Праграмны прынцып работы камп'ютара

Асноўным прынцыпам пабудовы ўсіх сучасных камп'ютараў з'яўляецца праграмнае кіраванне, у адпаведнасці з якім каманды праграмы і даныя захоўваюцца ў аператыўнай памяці ў закадзіраваным выглядзе. Інфармацыя, з якой працуе камп'ютар, прадстаўлена ў двайковым кодзе і падзяляецца на два тыпы: праграма (набор каманд па апрацоўцы даных); даныя, якія апрацоўваюцца праграмай. Працэсар можа выконваць арыфметычныя і лагічныя аперацыі, прадугледжаныя яго сістэмай каманд. Каманды і даныя счытваюцца па чарзе з памяці і паступаюць у працэсар, дзе яны расшыфроўваюцца, а затым выконваюцца. Вынікі выканання розных каманд могуць быць запісаны ў памяць ці перададзены на розныя ўстройствы.

Работа камп'ютара па прынцыпе праграмнага кіравання апісана ў прыкладзе 10.1. Стварэнне ЭВМ з праграмай, якая захоўваецца ў памяці, паклала пачатак праграміраванню, а магчымасць звароту да любой ячэйкі памяці па яе адрасе дазволіла выкарыстоўваць пераменныя ў праграміраванні.

10.2. Розныя падыходы да класіфікацыі праграмнага забеспячэння

Камп'ютар уяўляе сабой адзінства апаратных (hardware) і праграмных (software) сродкаў. З'яўленне перса-

нальнага камп'ютара і развіццё праграмiравання прывяло да ўзнікнення вялізнай колькасці розных праграм. Сукупнасць усіх праграмных сродкаў называюць **праграмным забеспячэннем (ПЗ)** камп'ютара.

Разгледзім некаторыя спосабы класіфікацыі ПЗ.

Класіфікацыя паводле прызначэння

У залежнасці ад прызначэння вылучаюць сістэмнае, прыкладное і інструментальнае ПЗ (прыклад 10.2). Кожны клас у сваю чаргу падзяляецца на падкласы. Падрабязную схему дзялення ПЗ можна паглядзець у *Дадатку да главы 2* (с. 116).

Класіфікацыя паводле спосабу распаўсюджвання і выкарыстання

Тып распаўсюджвання і выкарыстання праграмы залежыць ад ліцэнзіі. Ліцэнзія на праграмнае забеспячэнне — прававы інструмент, які вызначае выкарыстанне і распаўсюджванне праграмнага забеспячэння, ахаванага аўтарскім правам. Ліцэнзія выступае гарантыяй таго, што выдавец ПЗ, якому належаць выключныя правы на праграму, не падасць у суд на карыстальніка. Звычайна ліцэнзія на праграмнае забеспячэнне дазваляе атрымальніку выкарыстоўваць адну ці некалькі копій праграмы, прычым без ліцэнзіі такое выкарыстанне разглядаецца як парушэнне аўтарскіх правоў выдаўца.

Спосабы распаўсюджвання праграмных прадуктаў: камерцыйны, умоўна-бясплатны, бясплатны і пробны (прыклад 10.3). Акрамя таго,

Прыклад 10.2. Класы ПЗ у залежнасці ад прызначэння.

Сістэмнае ПЗ — сукупнасць праграм для забеспячэння работы камп'ютара і камп'ютарных сетак. Праграмы, якія ўваходзяць у склад сістэмнага ПЗ, дазваляюць карыстальніку ажыццяўляць кіраўніцтва і кантроль над работай камп'ютара і камп'ютарнай сеткі, а таксама забяспечваюць магчымасць выканання іншых праграм.

Прыкладное ПЗ — комплекс праграм для рашэння задач пэўнага класа прадметнай вобласці. Дадзены клас ПЗ з'яўляецца самым шматлікім, сюды ўваходзяць рэдактары, ЭСН, камп'ютарныя гульні і г. д.

Інструментальнае ПЗ прызначана для стварэння іншага праграмнага забеспячэння. Сюды залічваюць сістэмы праграмiравання, якія забяспечваюць распрацоўку праграм.

Класіфікацыя паводле спосабу выканання праграмы

У большай меры неабходна праграмісту, чым звычайнаму карыстальніку. Па гэтым крытэрыі праграмы падзяляюцца на праграмы, якія кампілююцца, і праграмы, якія інтэрпрэтуюцца.

Зыходны код у праграм, якія кампілююцца, пераўтвараецца кампілятарам у машынны код і запісваецца ў файл, з асаблівым загалоўкам і/ці расшырэннем. Аперацыйная сістэма ідэнтыфікуе такі файл як выконваемы.

У праграмах, якія інтэрпрэтуюцца, зыходны код паслядоўна выконваецца з дапамогай спецыяльнай праграмы-інтэрпрэтатара.

Прыклад 10.3. Класы ПЗ у залежнасці ад спосабу распаўсюджвання і выкарыстання.

Камерцыйныя праграмы (Commercial software) ствараюцца з мэтай атрымання прыбытку ад іх выкарыстання, напрыклад шляхам продажу.

Умоўна-бясплатныя праграмы (shareware) распаўсюджваюцца па прынцыпе «пратэсціруй, перш чым купіць». Выкарыстоўваць праграму можна на працягу невялікага тэрміну (2 тыдні або месяц). Пасля заканчэння дадзенага тэрміну карыстальнік абавязаны купіць яе ці спыніць выкарыстанне праграмы і выдаліць яе.

Бясплатныя праграмы (Freeware) — праграмнае забеспячэнне, ліцэнзійнае пагадненне якога не патрабуе якіх-небудзь выплат праваўладальніку. Ліцэнзія не дае карыстальніку права на мадыфікацыю праграмы.

Пробныя праграмы (Betaware) — звычайна папярэднія (тэставыя) бэта-версіі камерцыйнага ці некамерцыйнага ПЗ. Можна выкарыстоўваць бясплатна, але часта ўжыванне абмяжоўваецца перыядам тэсціравання або функцыянальнасцю праграмы.

адрозніваюць свабоднае і прапрыетарнае ПЗ. Свабоднае ПЗ распаўсюджваецца з зыходнымі кодамі і можа быць зменена карыстальнікам. У прапрыетарнага ПЗ усе правы (выкарыстанне, распаўсюджванне, мадыфікацыя) належаць стваральніку.

Класіфікацыя паводле ступені пераноснасці

Дазваляе вылучыць кросплатформенныя і платформазалежныя праграмы. Кросплатформенныя праграмы працуюць больш чым на адной апаратнай платформе і/ці аперацыйнай сістэме. Тыповым прыкладам з'яўляецца праграмнае забеспячэнне, прызначанае для работы ў аперацыйных сістэмах Linux і Windows адначасова. Платформазалежныя праграмы працуюць толькі ў тым асяроддзі, для якога створаны.

1. У чым сутнасць прынцыпу праграмнага кіравання?
2. Паводле якіх крытэрыяў можна класіфікаваць праграмнае забеспячэнне?
3. Назавіце асноўныя класы ПЗ паводле прызначэння.

Практыкаванні

1 Вызначыце, да якога класа праграмнага забеспячэння (паводле прызначэння) належаць пералічаныя праграмы.

1. Архіватар.
2. Тэкставы рэдактар.
3. PascalABC.
4. Windows.
5. Браўзер.
6. Бухгалтарская праграма.

2 Адкрыйце сайт http://pascalabc.net/. Перайдзіце ў раздзел **Лицензионное соглашение**. Да якога класа належыць ліцэнзія PascalABC?

3 Знайдзіце інфармацыю пра ліцэнзію праграмы Inkscape.

4 Знайдзіце інфармацыю пра кросплатформенныя праграмы: тэкставы рэдактар, графічны рэдактар, рэдактар электронных табліц.

§ 11. Уяўленне даных

11.1. Інфармацыя і даныя

З курса фізікі вам вядома, што фізічныя аб'екты ў нашым свеце знаходзяцца ў стане бесперапыннага руху і ва ўзаемадзеянні, якое суправаджаецца з'яўленнем сігналаў. Узаемадзеянне сігналаў з фізічнымі целамі можа змяняць уласцівасці цел. Змяненні, якія можна вымяраць ці рэгістраваць, утвараюць даныя. **Даныя** — зарэгістраваныя сігналы.

Даныя нясуць у сабе інфармацыю пра падзеі, якія адбыліся ў матэрыяльным свеце, паколькі яны адлюстроўваюць зарэгістраваныя сігналы, што ўзніклі ў выніку гэтых падзей. Аднак даныя не тоесныя інфармацыі (прыклады 11.1—11.3). Для чалавека інфармацыя — змест атрыманых ім паведамленняў. Пры атрыманні інфармацыі памяншаецца няпэўнасць ведаў. Веды вызначаюць паводзіны чалавека, дазваляюць яму прымаць рашэнні, будаваць адносіны з іншымі людзьмі.

Любая інфармацыя нематэрыяльная, яна не мае формы, памераў, масы. Такім чынам, для існавання і распаўсюджвання ў нашым матэрыяльным свеце яна павінна быць абавязкова звязана з якім-небудзь матэрыяльным аб'ектам — **носьбітам інфармацыі**.

Матэрыяльным носьбітам інфармацыі можа быць папера, метал, пластмаса, паветра, электрамагнітнае поле і інш. Сігналы таксама з'яўляюцца матэрыяльнымі носьбітамі інфармацыі. Захоўванне інфармацыі звязана з

Прыклад 11.1. Адкрыўшы кнігу з тэкстам на замежнай мове, чалавек атрымае даныя, але не атрымае інфармацыю, паколькі яму не вядомы спосаб пераўтварэння даных, запісаных з дапамогай невядомых знакаў у вядомыя яму паняцці.

Прыклад 11.2. У вас ёсць файл з данымі, але вы не ведаеце, у якой праграме ён быў створаны. У гэтым выпадку вы маеце даныя, але не зможаце атрымаць інфармацыю датуль, пакуль не ўстановіце адпаведную праграму.

Прыклад 11.3. За тысячагоддзі эвалюцыі пах не паддаваўся вядомым спосабам фіксацыі і перадачы інфармацыі. Зразумець ці ўявіць незнаёмы пах вельмі цяжка. Аднак чалавек атрымлівае інфармацыю, адчуўшы пах. Работы па атрыманні даных пра пах вядуцца, але пра канчатковы вынік пакуль гаварыць рана. На сённяшні дзень пах — інфармацыя, але не даныя.

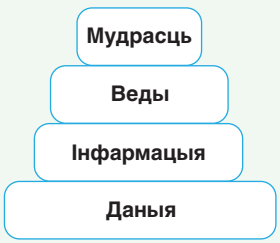
У 2013 г. вучоныя з Такійскага аграрна-тэхнічнага ўніверсітэта вынайшлі «экран, які можа мець пах». Пах зыходзіць з вобласці на экране, якая адпавядае крыніцы водару. Напрыклад, калі з'яўляецца малюнак персіка, адпаведны вугал экрана пахне фруктам.

На дадзены момант сістэма адначасова можа перадаваць толькі адзін пах.

Прыклад 11.4. Характарыстыкай носьбіта, якая не змяняецца з цягам часу, можа быць, напрыклад, намагнічанасць вобласці паверхні дыска ці літара на паперы. Характарыстыка носьбіта, якая змяняецца з цягам часу, — гэта, напрыклад, амплітуда ваганняў гукавой хвалі ці напружанне ў правадах.

Прыклад 11.5. Прыклады сігналаў: электрамагнітныя хвалі, змяненне электрычнага напружання, гукавая хваля, ваганні зямной кары, перадача даных па канале сувязі і інш.

У 1989 г. амерыканскі вучоны ў галіне даследавання аперацыі і тэорыі сістэм Расэл Акоф (1919—2009) прапанаваў іерархічную мадэль DIKW (англ. *data, information, knowledge, wisdom* — даныя, інфармацыя, веды, мудрасць).



Кожны ўзровень дабаўляе пэўныя ўласцівасці да папярэдняга:

- у аснове знаходзіцца ўзровень даных — знакі і сігналы;
- інфармацыя дабаўляе кантэкст — даныя падаюцца ў выглядзе фактаў, ідэй, тэорый;
- веды дабаўляюць механізм выкарыстання інфармацыі, вызначаюць, як чалавек будзе яе ўжываць;
- мудрасць дабаўляе ўмовы выкарыстання ведаў, накіраваныя на дасягненне пастаўленых мэт.

фіксацыяй стану носьбіта, а распаўсюджванне — з працэсам, які працякае ў носьбіце (прыклад 11.4).

Інфармацыя не існуе сама па сабе. Заўсёды маецца **крыніца**, якая перадае інфармацыю, і **прыёмнік**, які яе ўспрымае. У ролі крыніцы ці прыёмніка можа быць любы аб'ект матэрыяльнага свету: чалавек, устройства, жывёла, расліна. Гэта значыць, што інфармацыя заўсёды прызначана для пэўнага аб'екта.

Інфармацыя становіцца данымі тады, калі знаходзіцца спосаб зафіксаваць інфармацыю на матэрыяльным носьбіце з дапамогай якой-небудзь фармальнай мовы.

Даныя ператвараюцца ў інфармацыю толькі тады, калі імі зацікавіцца чалавек. Чалавек здабывае інфармацыю з даных, ацэньвае, аналізуе яе.

Дзеянні, якія выконваюцца над інфармацыяй, называюць **інфармацыйнымі працэсамі.** Да іх залічваюць працэсы атрымання, стварэння, збору, пошуку, апрацоўкі, запамінання, захоўвання, распаўсюджвання і выкарыстання інфармацыі.

У інфарматыцы паняцце «інфармацыя» часта атаясамліваецца з паняццем «даныя», паколькі асноўным інструментам для вывучэння і ажыццяўлення інфармацыйных працэсаў з'яўляюцца камп'ютарныя тэхналогіі. У якасці фармальнай мовы для ўяўлення даных у інфарматыцы з'яўляецца двайковы код. З дапамогай двайковага кода сёння можна прадстаўляць лікі, тэксты, відарысы, гук, відэа.

11.2. Аналагавае і лічбавае ўяўленне даных

Сігналы нясуць у сабе інфармацыю, паказаную ў выглядзе даных. Атрымліваючы значэнні сігналу, чалавек атрымлівае даныя, з якіх шляхам апрацоўкі здабываецца інфармацыя. Большасць сігналаў уяўляюць сабой фізічныя велічыні, якія змяняюцца ў часе (прыклад 11.5).

Сігнал можа быць прадстаўлены ў **аналагавай** (бесперапыннай) ці **дыскрэтнай**[1] форме.

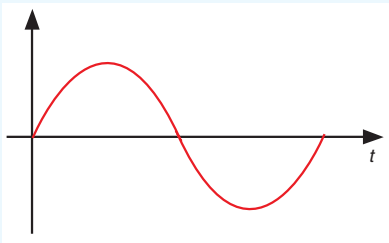
Аналагавы сігнал апісваецца функцыяй часу і бесперапынным мноствам магчымых значэнняў. **Дыскрэтны сігнал** з'яўляецца перарывістым (прыклады 11.6 і 11.7).

Дзякуючы сваім органам пачуццяў чалавек прывык мець справу з аналагавай інфармацыяй. Нашы зрок і слых, а таксама ўсе астатнія органы пачуццяў успрымаюць інфармацыю, што да нас паступае, у аналагавай форме, г. зн. бесперапынна ў часе.

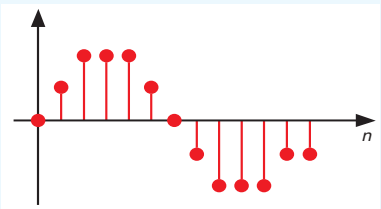
У камп'ютары інфармацыя пададзена ў лічбавым выглядзе. **Лічбавы сігнал** — сігнал, які можна паказаць у выглядзе паслядоўнасці лікавых значэнняў, запісаных з дапамогай лічбаў. У цяперашні час найбольш распаўсюджаны двайковыя лічбавыя сігналы. Гэта звязана з іх выкарыстаннем у камп'ютарных устройствах і прастатой кадзіравання.

Для атрымання лічбавага ўяўлення якога-небудзь аб'екта яго дыскрэтызуюць: атрымліваюць набор лікавых

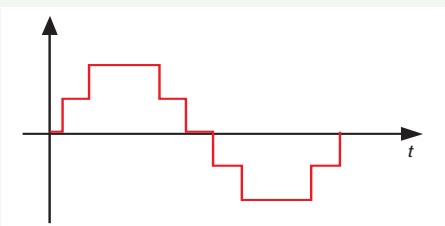
Прыклад 11.6. Аналагавы сігнал.



Прыклад 11.7. Дыскрэтны сігнал.



Каб паказаць аналагавы сігнал як паслядоўнасць лікаў, яго неабходна спачатку ператварыць у дыскрэтны сігнал, а затым выканаць квантаванне (сігнал, значэнні якога дыскрэтныя, а час бесперапынны).



У выніку сігнал будзе паказаны так, што на кожным прамежку часу апынецца вядома набліжанае (квантаванае) значэнне сігналу, якое можна запісаць лікам. Калі запісаць гэтыя цэлыя лікі ў двайковай сістэме, атрымаецца паслядоўнасць нулёў і адзінак, якая і будзе з'яўляцца лічбавым сігналам.

[1] Дыскрэтнасць — уласцівасць, якая супрацьпастаўляецца бесперапыннасці; перапыннасць.

Зыходнай велічынёй АЛП можа быць любая фізічная велічыня — напружанне, ток, супраціўленне, ёмістасць, частата паслядоўнасці імпульсаў, вугал павароту вала і інш.

Частата дыскрэтызацыі (або частата семпліравання, англ. *sample rate*) — частата ўзяцця адлікаў бесперапыннага па часе сігналу пры яго дыскрэтызацыі (вызначае, колькі разоў у секунду будзе вымераны зыходны сігнал). Вымяраецца ў герцах.

Прыклад 11.8. Сканеры.

Планшэтны сканер

Кніжны сканер

Сканер штрых-кода

Партатыўны сканер дакументаў

3D-сканер

Сканер кінаплёнкі

значэнняў, якія можна захаваць на электронным носьбіце. Гэтыя данныя з'яўляюцца лічбавай мадэллю аб'екта.

Працэс пераводу аналагавага ўяўлення аб'екта ў лічбавае называюць **алічбоўваннем (або аналага-лічбавым пераўтварэннем, АЛП).**

Алічбоўванне даных выконваецца на спецыяльным абсталяванні, якое дазваляе пераўтварыць аналагавы сігнал у лічбавы. Такія ўстройствы называюць аналага-лічбавымі пераўтваральнікамі.

У далейшым алічбаваныя даныя могуць выкарыстоўвацца для апрацоўкі на камп'ютары, перадачы па камп'ютарных сетках. Алічбаваць можна тэкст, фатаграфіі, малюнкі, гук, відэа, кіна- і фотаплёнкі.

Пры сканіраванні відарыса з фізічных аб'ектаў (тэкст, фатаграфіі, малюнкі) дыскрэтызацыя характарызуецца **разрашэннем** (колькасцю пікселяў на адзінку даўжыні па кожным з вымярэнняў) і **глыбінёй колеру.**

Для алічбоўвання тэксту ці графічных відарысаў ужываюцца розныя сканеры (прыклад 11.8). Сёння існуюць 3D-сканеры — устройствы, якія аналізуюць форму прадмета і ствараюць на аснове атрыманых даных яго 3D-мадэль. Сканеры кінаплёнкі дазваляюць пераўтварыць відарысы на кінаплёнцы ў лічбавыя відэафайлы. Праграмнае забеспячэнне для работы са сканерамі дазваляе наладжваць параметры сканіравання.

Пры вывадзе лічбавага відарыса на прынтар ці 3D-прынтар адбываецца адваротнае пераўтварэн-

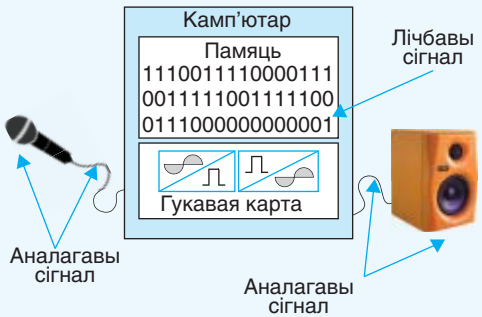
не — з лічбавай формы ў аналагавую. У выніку мы атрымліваем аналагавае ўяўленне аб'екта: малюнак на паперы ці матэрыяльны аб'ект.

Пры алічбоўванні сігналу, прывязанага да часу (гук, відэа), асноўнымі параметрамі з'яўляюцца **частата дыскрэтызацыі** (частата вымярэння) і **разраднасць** колькасці біт, якія вылучаюцца для запісу вынікаў вымярэння.

Гук у камп'ютар можна ўвесці з мікрафона ці з любога аўдыяўстройства, падключанага да камп'ютара. Аналага-лічбавы пераўтваральнік убудаваны ў гукавую карту. Алічбоўванне выконваецца спецыяльным праграмным забеспячэннем (напрыклад, Audacity). Пры вывадзе гуку адбываецца адваротнае пераўтварэнне сігналу з лічбавага ў аналагавы (прыклад 11.9). Для гэтага на гукавой карце маецца лічба-аналагавы пераўтваральнік.

У сучасныя смартфоны ўбудаваны лічбавы фотаапарат. Відарысы, атрыманыя з яго дапамогай, захоўваюцца ў лічбавай форме. Затым яны могуць быць загружаны ў камп'ютар для апрацоўкі, перадачы па вылічальных сетках ці для захоўвання. Відарысы можна прагледзець на экране манітора ці надрукаваць на прынтары.

Прыклад 11.9. Пераўтварэнне гуку:



1. У чым адрозненне інфармацыі і даных? Прывядзіце прыклады.
2. Што такое носьбіт інфармацыі?
3. Што разумеюць пад інфармацыйнымі працэсамі?
4. У чым адрозненне аналагавага сігналу ад лічбавага?
5. Што разумеюць пад алічбоўваннем?
6. Якія ўстройствы выкарыстоўваюць пры алічбоўванні?

Практыкаванні

1 Косця вучыцца ў мастацкай школе і піша карціны акварэллю. Мікіце спадабалася апошняя Косцева карціна, і ён сфатаграфаваў яе з дапамогай смартфона. Косця таксама вырашыў захаваць карціну ў лічбавым фармаце і адсканіраваў яе. Ці будуць аднолькавымі файлы ў Косці і Мікіты? Правядзіце сваё даследаванне па алічбоўванні відарысаў з дапамогай сканера і смартфона (лічбавага фотаапарата). Зрабіце вывады.

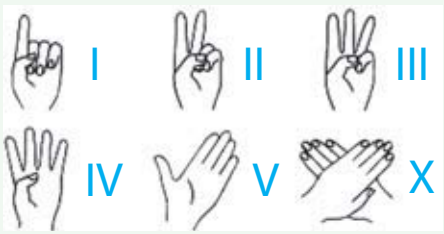
2 Падрыхтуйце паведамленні на адну з пералічаных тэм.
1. Лічбавы і аналагавы гук. Перавагі і недахопы.
2. Прававыя аспекты алічбоўвання кніг.
3. Тэхналогіі алічбоўвання відэа.

§ 12. Кадзіраванне лікавых даных

Прыклад 12.1. Спосабы запісу лікаў:

Сучасныя	1	2	3	4	10	20	30
Егіпецкія	I	II	III	IIII	∩	∩∩	∩∩∩
Грэчаскія (стары стыль)	I	II	III	IIII	Δ	ΔΔ	ΔΔΔ
Грэчаскія (новы стыль)	α	β	γ	δ	τ	κ	λ
Рымскія	I	II	III	IV	X	XX	XXX

У першабытнага чалавека прыладай лічэння былі пераважна пальцы. З іх дапамогай можна было лічыць да 5, а калі ўзяць дзве рукі, то і да 10. У старажытныя часы людзі хадзілі басанож. Таму яны маглі карыстацца для лічэння пальцамі як рук, так і ног.



Вядомыя народы, у якіх адзінкамі лічэння былі не пальцы, а іх фалангі.

Непазіцыйнай сістэмай лічэння з'яўляецца славянская, у якой замест лічбаў выкарыстоўваліся літары алфавіту. Каб адрозніваць літары ад лічбаў, над літарамі з лікавым значэннем пісаўся спецыяльны знак — цітла.

12.1. Паняцце сістэмы лічэння

Першыя камп'ютары называлі ЭВМ — электронна-вылічальная машына. Іх асноўным прызначэннем было выкананне разлікаў, для якіх неабходны лікавыя даныя. Існуе вялікая колькасць спосабаў уяўлення лікавых даных (прыклад 12.1). Са старажытных часоў людзі выкарыстоўвалі спецыяльныя значкі для абазначэння лікаў. Такія значкі называюць лічбамі.

Сістэма лічэння — спосаб запісу ліку з дапамогай набору ўмоўных знакаў, якія называюцца лічбамі.

Сістэмы лічэння бываюць пазіцыйнымі і непазіцыйнымі. У пазіцыйнай сістэме лічэння лікавае значэнне лічбы залежыць ад той пазіцыі, якую лічба займае ў запісе ліку. У непазіцыйнай сістэме лічэння лічба заўсёды мае адно і тое ж значэнне.

У цяперашні час чалавецтва выкарыстоўвае ў асноўным дзесятковую сістэму лічэння. У ёй для запісу лікаў выкарыстоўваецца 10 лічбаў: 0, 1, ...9. Лік 10 з'яўляецца асновай дзесятковай сістэмы лічэння.

Любы лік у дзесятковай сістэме лічэння можна запісаць як суму разрадных складаемых (прыклад 12.2). Лікі 1, 10, 100... з'яўляюцца разраднымі адзінкамі. Кожная разрадная адзінка можа быць запісана ў выглядзе 10^n.

Аналагічны запіс ліку можна атрымаць, калі замест 10 як асновы

сістэмы лічэння ўзяць адвольны лік p ($p > 1$). Разраднымі адзінкамі становяцца ступені асновы сістэмы лічэння. Для запісу ліку ў сістэме лічэння з асновай p спатрэбіцца p лічбаў. Звычайна выкарыстоўваюць першыя p лічбаў дзесятковай сістэмы лічэння: 0, 1, ..., (p – 1). Напрыклад, для чацвярычнай сістэмы лічэння гэта будуць лічбы 0, 1, 2, 3 (прыклад 12.3).

У агульным выглядзе лік Z можна запісаць наступным чынам:

$Zp = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p^1 + a_0 p^0$,

дзе лік p — аснова сістэмы лічэння, каэфіцыенты a_n, a_{n-1}, ... a_1, a_0 — лічбы ліку, значэнні p^n, p^{n-1}, ..., p^1, p^0 — разрадныя адзінкі.

Аснову сістэмы лічэння прынята паказваць як ніжні індэкс у дзесятковай сістэме. Напрыклад, дзесятковы лік 1443 можна запісаць як 1443_{10} або як $5A3_{16}$, 2643_8, 10110100011_2 (прыклад 12.4). Для дзесятковага ліку індэкс 10 можна не паказваць.

Дзесятковая сістэма лічэння з'яўляецца прыкладам пазіцыйнай сістэмы лічэння (прыклад 12.5). Прыкладам непазіцыйннай сістэмы лічэння з'яўляецца рымская.

У цяперашні час выкарыстоўваюцца пазіцыйныя сістэмы лічэння з асновамі 2, 3, 8, 10, 16. Пры рабоце з камп'ютарамі часцей за ўсё выкарыстоўваюцца шаснаццацярычная, васьмярычная, двайковая сістэмы лічэння.

Двайковая сістэма лічэння дазваляе запісваць лікі з дапамогай дзвюх лічбаў — 0 і 1. Запіс ліку ў двайковай сістэме лічэння з'яўляецца двайковым кодам ліку.

Прыклад 12.2. Запіс ліку 5973 у выглядзе сумы разрадных складаемых у дзесятковай сістэме лічэння:

$$5973 = 5 \cdot 1000 + 9 \cdot 100 + 7 \cdot 10 + 3 = 5 \cdot 10^3 + 9 \cdot 10^2 + 7 \cdot 10^1 + 3 \cdot 10^0.$$

Прыклад 12.3. Запіс ліку 12302_4 у выглядзе сумы разрадных складаемых у чацвярцічнай сістэме лічэння:

$$12302 = 1 \cdot 4^4 + 2 \cdot 4^3 + 3 \cdot 4^2 + 0 \cdot 4^1 + 2 \cdot 4^0.$$

Прыклад 12.4. Запіс ліку 1443_{10} у розных сістэмах лічэння:

$$5A3_{16} = 5 \cdot 16^2 + A \cdot 16^1 + 3 \cdot 16^0 = 5 \cdot 256 + 10 \cdot 16 + 3 = 1280 + 160 + 3 = 1443$$

$$2643_8 = 2 \cdot 8^3 + 6 \cdot 8^2 + 4 \cdot 8^1 + 3 \cdot 8^0 = 2 \cdot 512 + 6 \cdot 64 + 4 \cdot 8 + 3 = 1024 + 384 + 32 + 3 = 1443$$

$$10110100011_2 = 1 \cdot 2^{10} + 0 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 1024 + 0 + 1 \cdot 256 + 1 \cdot 128 + 0 + 1 \cdot 32 + 0 + 0 + 0 + 1 \cdot 2 + 1 = 1024 + 256 + 128 + 32 + 2 + 1 = 1443$$

Прыклад 12.5. Запіс ліку.

У запісе ліку 111 першая адзінка абазначае сотні, другая — дзясяткі, а трэцяя — адзінкі (лікавае значэнне — сто адзінаццаць).

У запісе ліку III кожная лічба I мае значэнне адзінкі (лікавае значэнне — тры).

З гісторыі вядома, што чалавек ужываў сістэмы лічэння з рознымі асновамі. У Кітаі доўга карысталіся пяцярычнай сістэмай лічэння. Плямёны мая лічылі ў дваццацярычнай сістэме лічэння. Шасцідзесяцярычную сістэму лічэння выкарыстоўвалі ў Вавілоне. Напамінам пра гэту сістэму лічэння сёння з'яўляецца дзяленне мінуты на 60 секунд, гадзіны — на 60 мінут, а вугла — на 360 градусаў.

У аснове лічэння тузінамі ляжыць дванаццацярычная сістэма лічэння, якая выкарыстоўваецца да гэтага часу: у годзе 12 месяцаў, на цыферблаце 12 гадзін. Для абазначэння лічбаў у дванаццацярычнай сістэме, акрамя 10 лічбаў дзесятковай сістэмы лічэння, выкарыстоўваліся яшчэ два значкі для абазначэння лікаў 10 і 11. У розныя часы і ў розных краінах выкарыстоўвалі: для 10 — T (англ. *ten*), D (лац. *decem*), X (рымскае 10); для 11 — E (англ. *eleven*) ці O (фр. *onze*). Можна выкарыстоўваць літары лацінскага алфавіта — A(10) і B(11). Акрамя таго, часам для абазначэння 10 выкарыстоўваюць перавернутую двойку (ƻ), для 11 — перавернутую тройку (ɛ).

Прыклад 12.6. Лікі ў шаснаццацярычнай сістэме лічэнння:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11 ... 19, 1A, 1B ... 1F, 20 ... 29, 2A ... 2F, 20 ... 99, 9A, 9B, 9C, 9D, 9E, 9F, A0, A1 ... FE, FF, 100...

Прыклад 12.7. Запіс лікаў у розных сістэмах лічэння.

Дзесятковая	Шаснаццацярычная	Двайковая
1	1	1
2	2	10
7	7	111
24	18	11 000
127	7F	1 111 111
256	100	100 000 000
1025	401	10 000 000 001

У васьмярычнай сістэме лічэння выкарыстоўваюцца лічбы ад 0 да 7. Яе ўжыванне для камп'ютараў абумоўлена тым фактам, што ў адным байце 8 біт. З дапамогай васьмярычных лікаў запісваюць коды лікаў і машынных каманд. Цяпер васьмярычную сістэму лічэння практычна цалкам выцесніла шаснаццацярычная.

У шаснаццацярычнай сістэме лічэння выкарыстоўваюцца 16 лічбаў: 10 лічбаў з дзесятковай сістэмы лічэння — 0...9 — і 6 літар лацінскага алфавіту — A, B, C, D, E, F (прыклад 12.6). Сістэма лічэння з асновай 16 шырока выкарыстоўваецца ў камп'ютарнай дакументацыі і пры напісанні праграм непасрэдна ў машынным кодзе. Напрыклад, для запісу адрасоў каманд, колеравых канстант.

12.2. Перавод лікаў з адной пазіцыйнай сістэмы лічэння ў іншую

Любы лік мае значэнне і форму ўяўлення. Значэнне ліку задае колькасную меру і вызначаецца яго адносінамі да значэнняў іншых лікаў («больш», «менш», «роўна»). Форма ўяўлення ліку вызначае спосаб запісу ліку з дапамогай прызначаных для гэтага лічбаў. Значэнне ліку не залежыць ад спосабу яго ўяўлення: лік з адным і тым жа значэннем можа быць запісаны па-рознаму (прыклад 12.7). Сістэмы лічэння вызначаюць форму ўяўлення лікаў, а паколькі іх шмат, то ўзнікае пытанне пра магчымасць і спосабы пераходу ад адной формы ўяўлення ліку да іншай.

У далейшым будзем разглядаць толькі пазіцыйныя сістэмы лічэння.

Перавод ліку з сістэмы лічэння з асновай p у сістэму лічэння з асновай q абазначаюць як $Z_p \to Z_q$. Непасрэдны перавод выконваць няпроста, таму часцей за ўсё разглядаюць пераводы $Z_p \to Z_r \to Z_q$, дзе звычайна $r = 10$. Гэта значыць для выканання пераводаў трэба ўмець пераводзіць лікі ў дзесятковую сістэму лічэння і з дзесятковай у сістэму лічэння з іншай асновай.

Для атрымання алгарытму пераводу ліку з дзесятковай сістэмы лічэння ў сістэму лічэння з іншай асновай разгледзім запіс ліку ў выглядзе сумы разрадных складаемых:

$$Z_p = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p^1 + a_0 p^0$$

У гэтай суме кожнае складаемае за выключэннем апошняга абавязкова дзеліцца на p. Тады атрымліваем, што апошняе складаемае a_0 з'яўляецца астачай ад дзялення зыходнага ліку на p. Падзелім лік на p, атрымаем суму разрадных складаемых са старшай ступенню, на 1 меншай. Знайшоўшы астачу яго дзялення на p, атрымаем значэнне a_1. Працягваючы такім чынам, атрымаем усе значэнні a_i.

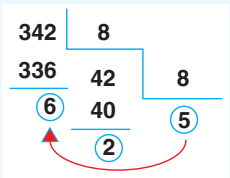
У прыкладзе 12.8 разгледжаны пераводы лікаў з дзесятковай сістэмы лічэння.

Алгарытм пераводу $Z_{10} \to Z_p$:

1. Падзяліць цалкам зыходны лік на аснову новай сістэмы лічэння p і знайсці астачу ад дзялення. Гэта будзе лічба a_0.

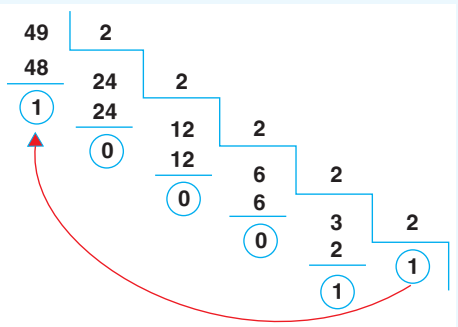
Прыклад 12.8. Перавод лікаў з дзесятковай сістэмы лічэння.

Дзеянні па алгарытме пераводу лікаў з дзесятковай сістэмы лічэння звычайна паказваюць «лесвічкай», г. зн. наступным чынам:



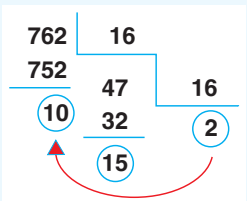
У прыкладзе рэалізаваны перавод ліку 342 у васьмярычную сістэму лічэння. Вынік — 526_8.

1. Перавесці лік 49 у двайковую сістэму лічэння:



$49 = 110001_2$

2. Перавесці лік 762 у шаснаццацярычную сістэму лічэння:



Для запісу выніку неабходна атрыманыя астачы прадставіць шаснаццацярычнымі лічбамі: $10 = A_{16}$, $15 = F_{16}$. Вынік $762 = 2FA_{16}$.

Прыклад 12.9. Перавод лікаў у дзесятковую сістэму лічэння:

$$11001_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 0 + 0 + 1 = 25_{10};$$

$$3045_8 = 3 \cdot 8^3 + 0 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 = 3 \cdot 512 + 0 + 32 + 5 = 1573_{10};$$

$$A3D_{16} = A \cdot 16^2 + 3 \cdot 16^1 + D \cdot 16^0 = 10 \cdot 256 + 3 \cdot 16 + 13 = 2621_{10}$$

Дробныя лікі можна пераводзіць аналагічна:

$$12{,}3_5 = 1 \cdot 5^1 + 2 \cdot 5^0 + 3 \cdot 5^{-1} = 5 + 2 + \frac{3}{5} = 7 + 0{,}6 = 7{,}6_{10}.$$

Прыклад 12.10. Табліца тэтрад, трыяд і двайковых пар.

Лічба	Тэтрада	Трыяда	Пара
0	0000	000	00
1	0001	001	01
2	0010	010	10
3	0011	011	11
4	0100	100	
5	0101	101	
6	0110	110	
7	0111	111	
8	1000		
9	1001		
A	1010		
B	1011		
C	1100		
D	1101		
E	1110		
F	1111		

Прыклад 12.11. Перавод ліку з шаснаццацярычнай сістэмы лічэння ў двайковую:

$$B27_{16} = \underbrace{B}_{1011}\ \underbrace{2}_{0010}\ \underbrace{7}_{0111} = 101100100111_2$$

2. Дзель ад дзялення зноў падзяліць цалкам на p з вылучэннем астачы. Працягваць да таго часу, пакуль дзель ад дзялення не стане меншай за p.

3. Атрыманыя астачы ад дзялення, запісаныя ў парадку, адваротным парадку іх атрымання, з'яўляюцца запісам ліку ў сістэме лічэння з асновай p.

Алгарытм пераводу $Z_p \to Z_{10}$ выцякае са спосабу ўяўлення ліку ў сістэме лічэння з асновай p:

1. Уявіць лік у выглядзе сумы разрадных складаемых па ступенях p.

2. Выканаць арыфметычныя аперацыі ў дзесятковай сістэме лічэння.

Асобна разглядаецца сітуацыя пераводаў $Z_p \to Z_q$, калі p і q з'яўляюцца ступенямі двойкі. У гэтым выпадку ў якасці прамежкавай сістэмы лічэння зручна выбіраць двайковую. Перавод з сістэмы лічэння з асновай ступені двойкі ў двайковую заснаваны на тым, што кожнай лічбе ў гэтай сістэме лічэння адпавядае група двайковых лічбаў:

- шаснаццацярычнай лічбе адпавядае група з чатырох двайковых лічбаў ($16 = 2^4$), якая называецца тэтрадай;
- васьмярычнай лічбе адпавядае група з трох двайковых лічбаў ($8 = 2^3$), якая называецца трыядай;
- чацвярычнай лічбе адпавядае пара двайковых лічбаў ($4 = 2^2$).

У прыкладзе 12.9 прыведзены пераводы лікаў у дзесятковую сістэму лічэння.

Табліцы тэтрад, трыяд і двайковых пар прыведзены ў прыкладзе 12.10. Для пераводу ліку з сістэмы лічэння з асновай 16 у двайковую сістэму

лічэння кожную лічбу ліку замяняюць адпаведнай тэтрадай (прыклад 12.11). Пры пераводзе з васьмярычнай сістэмы лічэння лічбы замяняюцца трыядамі (прыклад 12.12), а з чацвярычнай — парамі (прыклад 12.13).

Пры пераводзе з двайковай сістэмы лічэння лік разбіваецца адпаведна на групы па 4, 3 або 2 лічбы справа налева. Пры неабходнасці злева да ліку можна прыпісаць нулі. Затым ажыццяўляецца замена тэтрады (трыяды або пары) на адпаведную лічбу (прыклад 12.14). Перавод $Z_{16} \to Z_8$ і $Z_4 \to Z_{16}$ паказаны ў прыкладах 12.15 і 12.16.

Калькулятар у АС Windows дазваляе выканаць пераводы лікаў з адной сістэмы лічэння ў іншую. Працуе калькулятар з сістэмамі лічэнняў, асновамі якіх з'яўляюцца 2, 8, 10 і 16. Для ажыццяўлення пераводаў калькулятар павінен быць у рэжыме **Праграммист** (прыклад 12.17). Абазначэнні для сістэм лічэння: Hex — шаснаццацярычная, Dec — дзесятковая, Oct — васьмярычная, Bin — двайковая. Для пераводу ліку з дапамогай калькулятара трэба:

1. Выбраць аснову сістэмы лічэння зыходнага ліку.
2. Набраць лік.
3. Вынік адлюструецца адразу для ўсіх сістэм лічэння.

У рэжыме **Праграммист** калькулятар можа працаваць толькі з цэлымі лікамі.

Прыклад 12.12. Перавод ліку 362_8 у двайковую сістэму лічэння:

$$362_8 = \underbrace{3}_{011}\ \underbrace{6}_{110}\ \underbrace{2}_{010} = 011110010_2$$

Нуль у пачатку запісу ліку можна прапусціць. Адказ: 11110010_2.

Прыклад 12.13. Перавод ліку 3202_4 у двайковую сістэму лічэння:

$$3202_4 = \underbrace{3}_{11}\ \underbrace{2}_{10}\ \underbrace{0}_{00}\ \underbrace{2}_{10} = 11100010_2$$

Прыклад 12.14. Перавод лікаў з двайковай сістэмы лічэння (знак ′ адзяляе тэтрады, трыяды або пары).

У шаснаццацярычную:

$$110011010_2 = 0001\,'\,1001\,'\,1010 = 19A_{16}$$

У васьмярычную:

$$11011010111_2 = 01\,'\,011\,'\,010\,'\,111 = 3327_8$$

У чацвярычную:

$$10110111_2 = 10\,'\,11\,'\,01\,'\,11 = 2313_4$$

Прыклад 12.15. Перавод ліку $C36_{16}$ у васьмярычную сістэму лічэння.

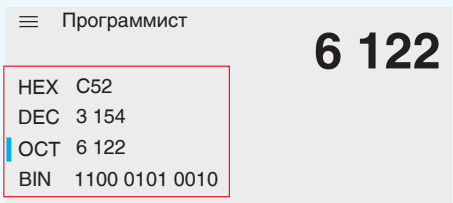
Спачатку перавядзём лік у двайковую сістэму лічэння, затым разаб'ём яго на трыяды і атрымаем васьмярычны запіс: $C56_{16} = 1100\ 0101\ 0110_2 = 110\,'\,001\,'\,010\,'\,110 = 6126_8$.

Прыклад 12.16. Перавод ліку 23103_4 у шаснаццацярычную сістэму лічэння.

Перавядзём лік у двайковую сістэму лічэння, затым разаб'ём яго на тэтрады і атрымаем шаснаццацярычны запіс:

$$23103_4 = 10\ 11\ 01\ 00\ 11_2 = 0010\,'\,1101\,'\,0011 = 2D3_{16}.$$

Прыклад 12.17. Перавод ліку 6122_8 з дапамогай калькулятара.



1. Што такое сістэма лічэння?
2. Якімі бываюць сістэмы лічэння?
3. Як перавесці лік у дзесятковую сістэму лічэння?
4. Як перавесці лік з дзесятковай сістэмы лічэння?
5. Для чаго выкарыстоўваюцца трыяды і тэтрады пры пераводзе лікаў з адной сістэмы лічэння ў іншую?

Практыкаванні

1 Перавядзіце лікі ў дзесятковую сістэму лічэння.

1. 1001_2, 1110101_2, 100001_2.
2. 2121_3, 2001_3, 2213_4, 2332_4.
3. 456_8, 302_8, 165_8.
4. $A54_{16}$, 679_{16}, FDC_{16}.

2 Перавядзіце лікі з дзесятковай сістэмы лічэння ў адзначаную.

1. $345, 219, 50270 \to Z_{16}$.
2. $234, 672, 1021 \to Z_8$.
3. $92, 131 \to Z_4$.
4. $85, 201 \to Z_3$.
5. $85, 129, 311 \to Z_2$.

3 Выкананне пераводу «лесвічкай» можна ажыццявіць у Excel. Адкрыйце файл з прыкладам 12.9. Неабходныя формулы можна паглядзець у рэжыме паказу формул (Ctrl + ~). Выкарыстайце прыклад для пераводу лікаў з практыкавання 2.

	A	B	C
1	342	8	
2	=B2*B1	=ЦЕЛОЕ(A1/B1)	8
3	=A1-A2	=C3*C2	=ЦЕЛОЕ(B2/C2)
4		=B2-B3	

4* Знайдзіце ў Excel даведку па функцыях ДЕС, ДВ.В.ДЕС, ВОСЬМ.В.ДЕС і інш. Выкарыстоўвайце гэтыя функцыі для праверкі правільнасці выканання пераводу лікаў у практыкаваннях 1 і 2.

5 Ажыццявіце перавод лікаў паміж пералічанымі сістэмамі лічэння:

1. $3201_4 \to Z_2 \to Z_8 \to Z_{16}$;
2. $5612_8 \to Z_2 \to Z_4 \to Z_{16}$;
3. $F1A_{16} \to Z_2 \to Z_4 \to Z_8$;
4. $4567_8 \to Z_{16} \to Z_4$;
5. $D91_{16} \to Z_8 \to Z_4$.

6 Вызначыце, у якім парадку трэба ажыццяўляць перавод ліку ў наступныя сістэмы лічэння: $Z_{10} \to Z_2 \to Z_8 \to Z_{16}$ і $Z_4 \to Z_{16} \to Z_{10}$. Адказ абгрунтуйце.

7 Запішыце мінімальны і максімальны.

1. Пяцізначныя лікі ў чацвярычнай сістэме лічэння.
2. Чатырохзначныя лікі ў васьмярычнай сістэме лічэння.
3*. Трохзначныя лікі ў дванаццацярычнай сістэме лічэння.

Атрымайце дзесятковыя ўяўленні запісаных лікаў. Зрабіце вывады.

8 Запішыце лік, які ідзе за 34_8, 22_3, 34_5, 1111_2, CF_{16}.

9 Запішыце лік, які папярэднічае 54_7, 30_4, 100_{12}.

10 Рашыце задачы.

1. Знайдзіце найменшы з лікаў A, B, C і D, запісаных у розных сістэмах лічэння, калі $A = 1023_4$, $B = 471_6$, $C = 69_{10}$, $D = 1001010_2$.
2. У сістэме лічэння з некаторай асновай p лік 58_{10} запісваецца як 213_p. Знайдзіце гэту аснову.
3. Вызначыце ўсе асновы сістэм лічэння, у якіх запіс ліку 17 заканчваецца на 2.
4. Да запісу натуральнага ліку ў васьмярычнай сістэме лічэння справа прыпісалі два нулі. У колькі разоў павялічыўся лік? Адказ запішыце ў дзесятковай сістэме лічэння.

5*. У садзе 100_p фруктовых дрэў. З іх 34_p яблыні, 25_p груш і 5_p вішань. Якая сістэма лічэння выкарыстоўваецца пры падліку колькасці дрэў?

11 Падрыхтуйце паведамленні на адну з пералічаных тэм.

1. Гісторыя лічэння.
2. Механічныя лічыльныя прыстасаванні.
3. Выкарыстанне траічнай сістэмы лічэння.

§ 13. Кадзіраванне тэкставых даных

13.1. Уяўленне тэксту

Натуральнай для органаў пачуццяў чалавека з'яўляецца аналагавая форма ўяўлення інфармацыі, аднак дыскрэтная форма ўяўлення з дапамогай некаторага набору знакаў найбольш універсальная. Для запісу тэксту выкарыстоўваюць сімвалы алфавіту (прыклад 13.1).

Уяўленне інфармацыі ў алфавітнай (тэкставай) форме — самы распаўсюджаны спосаб з часоў вынаходства пісьменства. Інфармацыя перадаецца ў выглядзе тэксту, запісанага на якой-небудзь мове: рускай, беларускай і г. д. Для запісу тэксту на розных мовах можна выкарыстоўваць адзін алфавіт. Напрыклад, для запісу тэксту на рускай ці беларускай мовах выкарыстоўваюць кірыліцу, а для запісу тэксту на англійскай ці нямецкай мовах — лацінку.

Прыклад 13.1. Розныя алфавіты.

Аа	Бб	Вв	Гг	Дд	Ее	Ёё	Жж	Зз
Ии	Йй	Кк	Лл	Мм	Нн	Оо	Пп	Рр
Сс	Тт	Уу	Фф	Хх	Цц	Чч	Шш	Щщ
Ъъ	Ыы	Ьь	Ээ	Юю	Яя			

Кірылічны алфавіт

Aa	Bb	Cc	Dd	Ee	Ff	Gg	Hh	Ii
Jj	Kk	Ll	Mm	Nn	Oo	Pp	Qq	Rr
Ss	Tt	Uu	Vv	Ww	Xx	Yy	Zz	

Лацінскі алфавіт

Αα	Ββ	Γγ	Δδ	Εε	Ζζ	Ηη	Θθ
Ιι	Κκ	Λλ	Μμ	Νν	Ξξ	Οο	Ππ
Ρρ	Σσ	Ττ	Υυ	Φφ	Χχ	Ψψ	Ωω

Грэчаскі алфавіт

Прыклад 13.2. Кадзіраванне сімвалаў:

A	0100 0001
S	0101 0011
C	0100 0011
/	0010 1111
8	0011 1000
t	0111 0100
e	0110 0101

Прыклад 13.3. Фрагменты кодавых табліц: 1 — шаснаццацярычныя коды сімвалаў; 2 — дзесятковыя коды сімвалаў.

①

sp 20	! 21	« 22	# 23	$ 24	% 25	& 26	‘ 27
(28	) 29	* 2A	+ 2B	, 2C	- 2D	. 2E	/ 2F
0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37
8 38	9 39	: 3A	; 3B	< 3C	= 3D	> 3E	? 3F

②

sp 32	! 33	« 34	# 35	$ 36	% 37	& 38	‘ 39	(40	) 41
* 42	+ 43	, 44	- 45	. 46	/ 47	0 48	1 49	2 50	3 51
4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61

Прыклад 13.4. Кадзіраванне літары «Л» (рускай) у розных 8-разрадных кодавых табліцах.

Назва табліцы	Код	Дзесятковае значэнне
Windows-1251	11001011	203
КОИ-8	11101100	236
CP855	11010001	209
CP866	10001011	139

Прыклад 13.5. Тэкст у розных кадзіроўках.

Windows 1251

Пример текста в разных кодировках

КОИ-8

оПХЛЕП РЕЙЯРЮ Б ПЮГМШУ ЙНДХПНБЙЮУ

CP866

╧ЕшьхЕ Єхъеєр т Ерчэ√ї ъюфшЕютърї

Для запісу тэксту ў памяць камп'ютара выкарыстоўваюць двайковы код — алфавіт з двух сімвалаў: 0 і 1.

13.2. Паняцце кодавай табліцы

Тэкставая інфармацыя складаецца з сімвалаў: літар, лічбаў, знакаў прыпынку і інш. Мноства гэтых сімвалаў утварае камп'ютарны алфавіт. Тэкст, які складаецца з дадзеных сімвалаў, чалавек бачыць на экране манітора.

Камп'ютар можа апрацоўваць інфармацыю толькі ў лікавай форме, прадстаўленай у выглядзе двайковага кода. Таму для кадзіравання тэксту кожнаму сімвалу алфавіта ставяць у адпаведнасць двайковы код (прыклад 13.2). Часта ўсім знакам алфавіта ставяцца ў адпаведнасць коды, якія змяшчаюць аднолькавы лік двайковых разрадаў.

Сукупнасць усіх сімвалаў камп'ютарнага алфавіта і адпаведных ім двайковых кодаў запісваюць у выглядзе табліцы. Такую табліцу называюць **кодавай (кадзіровачнай) табліцай сімвалаў.**

З дапамогай кодавых табліц выконваюць кадзіраванне і дэкадзіраванне тэксту. Часта для зручнасці карыстальніка ў кодавых табліцах замест двайковага кода запісваецца яго дзесятковы ці шаснаццацярычны аналаг (прыклад 13.3). Для атрымання двайковага кода (з дзесятковага ці шаснаццацярычнага) трэба ажыццявіць пераклад ліку ў двайковую сістэму лічэння, для атрымання шаснаццацярычнага кода — у шаснаццацярычную сістэму лічэння.

Для розных камп'ютарных сістэм могуць выкарыстоўвацца розныя кода-

выя табліцы сімвалаў. У розных кодавых табліцах адным і тым жа сімвалам ставіцца ў адпаведнасць розны двайковы код (прыклад 13.4). У гэтым выпадку тэкст, створаны на адным камп'ютары, нельга будзе прачытаць на іншым камп'ютары без дадатковага перакадзіравання — сімвалы будуць адлюстроўвацца некарэктна (прыклад 13.5). Міжнародным стандартам стала табліца кадзіроўкі ASCII (*American Standard Code for Information Interchange* — амерыканскі стандартны код для абмену інфармацыяй). Дадзеная табліца падтрымлівае 8-разрадны двайковы код. Гэта значыць, што кожны сімвал будзе закадзіраваны паслядоўнасцю з 8 нулёў і адзінак. Такая паслядоўнасць і будзе кодам сімвала. Усяго ў табліцы $2^8 = 256$ знакаў.

Паколькі кожны сімвал кадзіруюць паслядоўнасцю з 8 нулёў і адзінак, ён займае ў памяці камп'ютара 8 біт (1 байт). У прыкладах 13.6—13.9 паказана, як з дапамогай табліцы сімвалаў ASCII (гл. *Дадатак да главы 2*, с. 118—119) кадзіраваць і дэкадзіраваць сімвалы.

У табліцы ASCII сімвалы лацінскага і рускага алфавітаў (вялікія і маленькія) ідуць па алфавіце. Дзесятковыя лічбы размешчаны ў парадку ўзрастання іх лікавых значэнняў. Гэта правіла звычайна выконваецца і ў іншых кодавых табліцах. Такі спосаб кадзіравання тэксту дазваляе сартаваць тэкставыя даныя ў алфавітным парадку, а лікавыя — па ўзрастанні іх значэнняў.

У кодавай табліцы ASCII захоўваецца 256 сімвалаў. Калі трэба працаваць з тэкстамі адразу на некалькіх мовах, то гэтых сімвалаў недастаткова.

Табліца ASCII кодаў складаецца з дзвюх частак. Першая частка **(стандартная)** змяшчае лацінскія літары, лічбы, прабел, знакі прыпынку і спецыяльныя сімвалы: +, /, *, %, # і інш. Сімвалы гэтай часткі маюць коды ад 00000000 да 01111111 (дзесятковыя аналагі 0—127). Другую частку кодавай табліцы называюць **альтэрнатыўнай**. Сімвалы ў ёй кадзіруюцца значэннямі ад 10000000 да 11111111 (дзесятковыя аналагі 128—255). Гэта частка табліцы выкарыстоўваецца для кадзіравання сімвалаў нацыянальных алфавітаў і сімвалаў псеўдаграфікі, якія выкарыстоўваюцца для адлюстравання рамак і ліній (сімвалы з кодамі 176—223). У стандартнай частцы кодавай табліцы коды ўсіх сімвалаў пачынаюцца з 0, у альтэрнатыўнай — з 1.

Прыклад 13.6. Азначэнне кода сімвала «@».

Нумар сімвала «@» у кодавай табліцы 64. Перавядзём лік 64 з дзесятковай сістэмы лічэння ў двайковую. Атрымаем: 1000000. Для атрымання 8-бітнага кода дабавім 0 перад лікам. 01000000 — двайковы код сімвала «@».

Прыклад 13.7. Азначэнне кода сімвала «Ф».

Слева ад літары «Ф» лік 148 — яе дзесятковы код. Перавядзём лік 148 у двайковую сістэму лічэння. Атрымаем 10010100 — двайковы код літары «Ф».

Прыклад 13.8. Азначэнне для сімвала «Z» шаснаццацярычнага кода.

Знойдзем у табліцы літару «Z». Побач з ёй лік 90. Перавядзём дзесятковы лік 90 у шаснаццацярычную сістэму лічэння. Атрымаем: 5A.

Прыклад 13.9. Дэкадзіраванне паслядоўнасці кодаў сімвалаў.

10101000 10101101 11100100 10101110 11100000 10101100 10100000 11100010 10101000 10101010 10100000

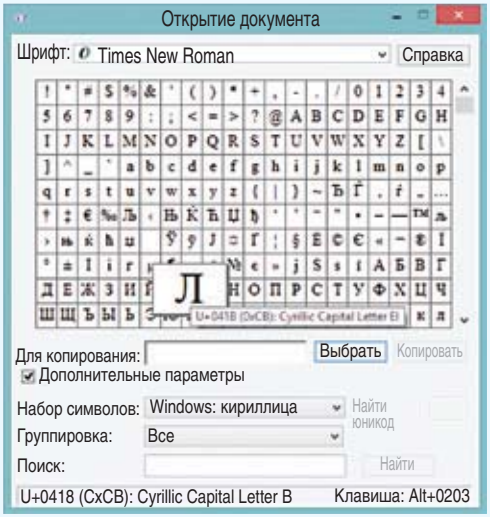
Заменім кожны двайковы код яго дзесятковым аналагам і атрымаем:

168 173 228 174 224 172 160 226 168 170 160

Цяпер у кодавай табліцы знойдзем адпаведныя сімвалы і атрымаем:

информатика

Прыклад 13.10. Табліца сімвалаў (фрагмент).



Прыклад 13.12. Кадзіраванне рускага слова «диск».

У кадзіроўцы ASCII слову «диск» адпавядае паслядоўнасць шаснаццацярычных кодаў: A4 A8 E1 AA (у дзесятковым уяўленні: 164 189 225 170). Гэтай паслядоўнасці кодаў у кадзіроўцы Unicode будуць адпавядаць сімвалы: ¤¨á ª. У кадзіроўцы Unicode слова «диск» будзе закадзіравана паслядоўнасцю шаснаццацярычных кодаў: 0434 0438 0441 043A.

Фармат UTF-8 быў распрацаваны 2 верасня 1992 г. Кенам Томпсанам і Робам Пайкам і рэалізаваны ў АС Plan 9 (аперацыйная сістэма, распрацаваная Bell Labs — зараз падраздзяленне Nokia — у канцы 1980-х гг. з разлікам на сеткі і рабочыя станцыі). Гэта кадзіроўка знайшла шырокае выкарыстанне ў UNIX-падобных АС.

Цяпер шырока выкарыстоўваюць кадзіроўку Unicode (Юнікод). У ёй камп'ютарны алфавіт складаецца не з 256, а з 65 536 сімвалаў. Для кадзіравання аднаго сімвала выкарыстоўваецца паслядоўнасць з 0 і 1, якая мае даўжыню 16 сімвалаў. Пры такой кадзіроўцы кожны сімвал будзе займаць у памяці камп'ютара 2 байты. Прагледзець кодавую табліцу на вашым камп'ютары можна запусціўшы праграму **Таблица символов** (знаходзіцца ў раздзеле **Стандартные** → → **Служебные** → **Таблица символов**).

Для вызначэння кода сімвала трэба выбраць гэты сімвал у табліцы (прыклад 13.10). На падказцы, якая ўсплыве, і ў ніжняй частцы акна будзе паказаны код сімвала ў шаснаццацярычнай сістэме лічэння і назва дадзенага сімвала (на англійскай мове).

Для пошуку сімвала, які адпавядае якому-небудзь коду, трэба ўвесці шаснаццацярычны код сімвала ў поле **Поиск**. У выпадаючым спісе **Набор символов** можна выбраць пэўны алфавіт (прыклад 13.11).

Стандартная частка кодавай табліцы ASCII супадае з пачаткам табліцы кадзіроўкі Unicode. Таму тэксты, якія змяшчаюць сімвалы, што размешчаны ў стандартнай частцы кодавай табліцы ASCII (лічбы, літары англійскага алфавіта), будуць без вялікіх намаганняў чытацца і ў кадзіроўцы Unicode.

Рускія сімвалы ў табліцы ASCII маюць коды, пачынаючы з ліку 80_{16} = = 128_{10}, а ў табліцы Unicode — з шаснаццацярычнага ліку 0410 (код вялікай літары А). Тэксты на рускай мове, набраныя ў кадзіроўцы ASCII, будуць няправільна адлюстроўвацца пры

праглядзе ў Unicode (прыклад 13.12). Для правільнага прагляду тэкст неабходна пераўтварыць.

Распаўсюджана кадзіроўка UTF-8 (англ. *Unicode Transformation Format, 8-bit* — фармат пераўтварэння Юнікода, 8 біт). Яна дазваляе больш кампактна захоўваць і перадаваць сімвалы, выкарыстоўваючы зменную колькасць байт (ад 1 да 4) для кадзіравання. Стандарт UTF-8 цяпер з'яўляецца самым распаўсюджаным у Інтэрнэце. Лацінскія літары, лічбы і найбольш распаўсюджаныя знакі прыпынку кадзіруюцца ва UTF-8 адным байтам, і коды гэтых сімвалаў адпавядаюць іх кодам у ASCII. Кірылічныя сімвалы кадзіруюцца двума байтамі (прыклад 13.13). Структуру двайковага кода сімвала ў кадзіроўцы UTF-8 можна паглядзець у *Дадатку да главы 2* (с. 117).

Тэксты ўводзяцца ў памяць камп'ютара ў асноўным з дапамогай клавіятуры. На клавішах напісаны знаёмыя нам літары, лічбы, знакі прыпынку і іншыя сімвалы. Націсканне на пэўную клавішу кадзіруе сімвал, і ў памяці камп'ютара ён захоўваецца ў форме двайковага кода. Пры выводзе сімвала на экран манітора знешні выгляд сімвала аднаўляецца па яго двайковым кодзе. Кадзіраванне і дэкадзіраванне тэкставых даных адбываецца таксама пры запісе тэксту ў файл на камп'ютарны носьбіт і пры счытванні тэксту з файла (прыклад 13.13).

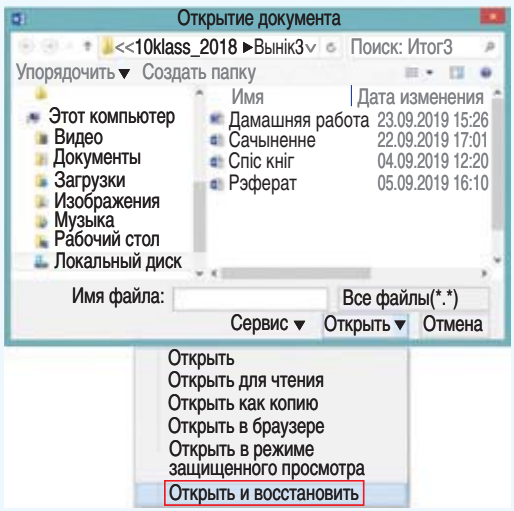
Інфармацыйны аб'ём тэксту залежыць ад колькасці сімвалаў у ім і спосабу кадзіравання сімвалаў. Інфармацыйная вага аднаго сімвала роўна 1 байту пры выкарыстанні аднабайтных

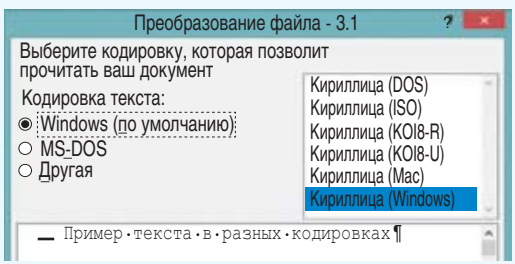
Шмат якія браўзеры забяспечваюць карыстальніку магчымасць выбару кадзіроўкі старонкі сайта. Аднак распаўсюджванне кадзіроўкі UTF-8 прыводзіць да таго, што гэта функцыя становіцца малазапатрабаванай. Па гэтай прычыне распрацоўшчыкі Google Chrome знялі дадзеную функцыю з апошніх версій браўзера.

Прыклад 13.13. Кадзіраванне сімвалаў ва UTF-8 і Unicode.

Сімвал	UTF-8	Unicode
L	4C (1 байт)	00 4C
Л	D0 9B (2 байты)	04 1B

Пераўтварэнне кадзіроўкі тэкставага дакумента можа ажыццяўляцца пры яго адкрыцці ў Word:





Прыклад 13.14.

У кадзіроўцы Unicode кожны сімвал кадзіруецца 2 байтамі. Трэба падлічыць колькасць сімвалаў у сказе і памножыць на 2. У сказе 38 сімвалаў. Інфармацыйны аб'ём — $38 \cdot 2 = 76$ байт.

Прыклад 13.15.

У кадзіроўцы Unicode кожны сімвал кадзіруецца 2 байтамі, у кадзіроўцы ASCII — 1 байтам. Пры перакадзіроўцы інфармацыйны аб'ём паменшыўся ў 2 разы. Значыць, зыходны памер паведамлення — $12 \cdot 2 = 24$ байты, $24 \cdot 8 = 192$ біт.

Прыклад 13.16.

Інфармацыйны аб'ём паведамлення 8,5 Кбайт = $8{,}5 \cdot 1024 = 8704$ байты. Аднаму сімвалу паведамлення адпавядае 1 байт = 8 біт. З дапамогай 8 біт можна закадзіраваць $2^8 = 256$ сімвалаў.

Прыклад 13.17*.

У Unicode кожны сімвал кадзіруецца 2 байтамі, таму ў паведамленні $(21 \cdot 1024)/2 = 10\,752$ сімвалаў. Калі б усе сімвалы ў паведамленні былі лацінскімі, то інфармацыйны аб'ём паведамлення ў кадзіроўцы UTF-8 склаў бы 10 752 байты. Інфармацыйны аб'ём паведамлення да перакадзіравання — 15 Кбайт = $15 \cdot 1024 = 15\,360$ байт. $15\,360 - 10\,752 = 4608$ — колькасць сімвалаў рускага алфавіта.

Прыклад 13.18.

На адной старонцы $30 \cdot 45 = 1350$ сімвалаў. У кодавай табліцы Unicode адзін сімвал кадзіруецца 2 байтамі. Інфармацыйны аб'ём адной старонкі — $1350 \cdot 2 = 2700$ байт. Усё апавяданне займае $80 \cdot 1024 = 81\,920$ байт. Тады колькасць старонак складае $81\,920/2700 \approx \approx 30{,}34 = 31$ старонка.

(8-бітных) кодавых табліц і 2 байтам пры выкарыстанні табліцы Unicode. Пры выкарыстанні табліцы UTF-8 інфармацыйная вага аднаго сімвала можа складаць ад 1 да 4 байт.

13.3. Рашэнне задач на кадзіраванне тэксту

Прыклад 13.14. Вызначыць інфармацыйны аб'ём наступнага сказа на рускай мове, калі яго закадзіравалі з дапамогай кодавай табліцы Unicode:

Программирование — вторая грамотность.

Прыклад 13.15. Аўтаматычнае ўстройства ажыццявіла перакадзіроўку паведамлення з кадзіроўкі Unicode у кадзіроўку ASCII. Пры гэтым інфармацыйны аб'ём паведамлення паменшыўся на 12 байт. Колькі біт было ў першапачатковым паведамленні?

Прыклад 13.16. Інфармацыйны аб'ём паведамлення 8,5 Кбайт. Дадзенае паведамленне змяшчае 8704 сімвалы. Якая максімальна магчымая колькасць сімвалаў змяшчаецца ў алфавіце?

Прыклад 13.17*. Аўтаматычнае ўстройства ажыццявіла перакадзіроўку паведамлення, якое змяшчае сімвалы рускага і лацінскага алфавітаў з кадзіроўкі UTF-8 у 16-бітны Unicode. (Сімвалы лацінскага алфавіта кадзіруюцца адным байтам, а рускага — двума байтамі.) У выніку пераўтварэння паведамленне стала займаць 21 Кбайт замест першапачатковых 15 Кбайт. Колькі ў паведамленні сімвалаў рускага алфавіта?

Прыклад 13.18. Тэкст апавядання займае 80 Кбайт. На адной старонцы 30 радкоў па 45 сімвалаў. Кожны сімвал кадзіруецца 16 бітамі ў фармаце Unicode. Колькі старонак у апавяданні?

1. Што такое алфавіт?
2. Як кадзіруюцца сімвалы?
3. Чым адрозніваецца кадзіраванне тэксту пры выкарыстанні розных кодавых табліц?
4. Як вызначыць код сімвала ў дадатку **Таблица символов**?

Практыкаванні

1 Выкарыстоўваючы кодавую табліцу ASCII або Unicode (праграма **Таблица символов**), закадзіруйце наступныя тэкставыя даныя:

Файл	Байт
Кадзіраванне	Disk
Printer	Bit
Сістэма лічэння	

2 Дэкадзіруйце двайковы код, выкарыстоўваючы кодавую табліцу ASCII.

11101000 10101010 10101110 10101011 10100000

3 Дэкадзіруйце інфармацыю, выкарыстоўваючы табліцу ASCII.

172 174 164 165 172 (дзесятковыя лікі).
E1 AA A0 AD E0 (шаснаццацярычныя лікі).

4 Выкарыстоўваючы праграму **Таблица символов**, вызначыце коды сімвалаў $^1/_2$, ±, $, Щ, Ў.

5 Вызначыце інфармацыйны аб'ём паведамлення «Удзельнік алімпіяды можа пісаць праграмы на мовах праграміравання Pascal, Python або C++».

1. У кадзіроўцы ASCII.
2. У кадзіроўцы Unicode.
3*. У кадзіроўцы UTF-8.

6 Паведамленне, інфармацыйны аб'ём якога ў 16-бітнай кадзіроўцы роўны 480 байт, перакадзіравалі ў 8-бітную кадзіроўку. Пасля гэтага да паведамлення дапісалі некалькі сімвалаў, і яго інфармацыйны аб'ём стаў роўны 520 байт. Колькі сімвалаў дапісалі ў паведамленне?

7 Алфавіт племені Тумба-Юмба складаецца з 8 літар. Які інфармацыйны аб'ём адной літары?

8 Паведамленне, запісанае літарамі з 16-літарнага алфавіта, змяшчае 21 сімвал. Які інфармацыйны аб'ём паведамлення?

9 Артыкул, набраны на камп'ютары, змяшчае 6 старонак. На кожнай старонцы аднолькавая колькасць радкоў па 56 сімвалаў у радку. Інфармацыйны аб'ём артыкула 504 Кбіт. Вызначыце колькасць радкоў на кожнай старонцы тэксту, лічачы, што кожны сімвал закадзіраваны з выкарыстаннем Unicode.

10 Скорасць чытання навучэнца 10-га класа складае ў сярэднім 1024 сімвалы ў мінуту. Які інфармацыйны аб'ём атрымае навучэнец, калі будзе бесперапынна чытаць на працягу 30 мін тэкст, набраны на камп'ютары ў кадзіроўцы Unicode?

11 Для атрымання гадавой адзнакі па геаграфіі вучню патрабавалася напісаць рэферат на 15 старонак. Ён выканаў гэта заданне на камп'ютары, набіраючы тэкст у кадзіроўцы Unicode. Які аб'ём памяці (у Кбайтах) зойме рэферат, калі ў кожным радку па 72 сімвалы, а на кожнай старонцы змяшчаецца 28 радкоў? Кожны сімвал займае 2 байты памяці.

12 Ацаніце інфармацыйны аб'ём старонкі тэксту з вучэбнага дапаможніка па інфарматыцы. Для гэтага палічыце колькасць радкоў на старонцы і колькасць сімвалаў у радку. Для тэксту на белым і блакітным фоне разлікі трэба праводзіць асобна, а затым падсумаваць вынікі. Тэкст набраны з выкарыстаннем кадзіроўкі Unicode.

13 Пеця і Вася пішуць адзін аднаму лісты, кадзіруючы інфармацыю наступным чынам: кожны сімвал ліста кадзіруецца двайковым кодам па табліцы ASCII. Затым 0 замяняецца на 1, а 1 на 0. Па атрыманых кодах у табліцы адшукваюцца сімвалы, з якіх складаецца тэкст ліста. Той, хто атрымаў ліст, выконвае тыя ж дзеянні, каб ліст прачытаць. Напрыклад, для кадзіравання рускага слова «Привет» трэба паступіць так:

П – 143 – 10001111 – 01110000 – 112 – p
р – 224 – 11100000 –00011111 – 31 – ▼
и – 168 –10101000 – 01010111 – 87 – W
в – 162 – 10100010 – 01011101 – 93 –]
е – 165 – 10100101 – 01011010 – 90 – Z
т – 226 – 11100010 – 00011101 – 29 – ↔

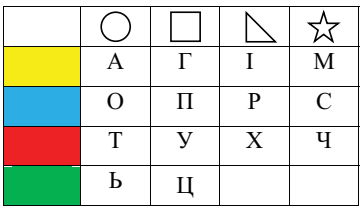
Атрымаем: p ▼ W] Z ↔

З дапамогай праграмы **Калькулятор** можна не толькі пераводзіць лікі ў двайковую сістэму лічэння, але і ажыццяўляць замену 0 на 1, а 1 на 0. Каб замяніць 0 на 1, а 1 на 0 на **Калькуляторе** (в рэжыме **Программист**), трэба выканаць дзеянне Xor над двума двайковымі лікамі: зыходным лікам і лікам 11111111 (напрыклад, 10001111 Xor 11111111 = 1110000).

Закадзіруйце гэтым спосабам: Привет, Вася! Как дела?

Дэкадзіруйце: p ▼ W] Z ↔ ▌■ } ▲ Z ■ ▒ Ф ▌

14 Для сакрэтнай перапіскі Оля і Света прыдумалі сваю кодавую табліцу. Дэкадзіруйце паведамленне ад Олі да Светы, выкарыстоўваючы частку табліцы. Прыдумайце коды для іншых літар рускага алфавіта.

Табліца	Паведамленне
	

15 Пацвердзіце ці абвергніце сцверджанне «СМС-паведамленне, набранае транслітам, будзе каштаваць танней, чым аналагічнае паведамленне, набранае рускімі літарамі».

§ 14. Кадзіраванне графікі, гуку і відэа

14.1. Кадзіраванне графікі

У цяперашні час пры стварэнні і захоўванні графічных аб'ектаў у камп'ютары выкарыстоўваюцца **растравы** і **вектарны** відарысы (прыклады 14.1, 14.2).

Растравы відарыс — сукупнасць асобных пунктаў (пікселяў), кожны з якіх мае свой колер.

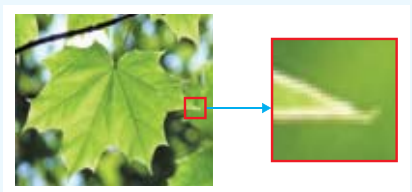
У вектарным графічным відарысе кожны намаляваны элемент з'яўляецца аб'ектам: лінія, авал, прамавугольнік і інш. Усе аб'екты маюць пэўны пералік значэнняў уласцівасцей, якія апісваюць гэтыя аб'екты (прыклад 14.3).

Вектарны відарыс — сукупнасць графічных прымітываў (аб'ектаў малюнка), якія апісаны з дапамогай лікавых значэнняў ці матэматычных формул.

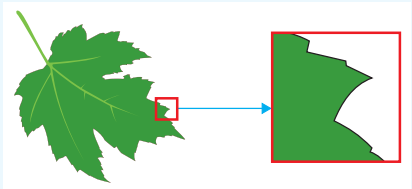
Адрозненне ва ўяўленні растравых і вектарных відарысаў існуе толькі для графічных файлаў. Пры вывадзе на экран манітора відарыса растравага ці вектарнага тыпаў у відэапамяці камп'ютара *фарміруецца інфармацыя растравага тыпу*. Гэта інфармацыя складаецца з двайковых кодаў кожнага пікселя. Код пікселя — інфармацыя пра яго колер.

Калі на чорна-белы відарыс накласці сетку і кожнай ячэйцы белага колеру паставіць у адпаведнасць 1, а чорнага колеру — 0 (ці наадварот: 1 — чорны, 0 — белы), то можна стварыць

Прыклад 14.1. Растравы відарыс.



Прыклад 14.2. Вектарны відарыс.



Прыклад 14.3. Уласцівасці адрэзка і круга ў вектарным відарысе.

Адрэзак:

- Пачатак і канец адрэзка — дзве пары лікаў, якія вызначаюць каардынаты пунктаў на каардынатнай плоскасці;
- значэнні, якія вызначаюць колер, таўшчыню і тып лініі (суцэльная, пункцірная і інш.).

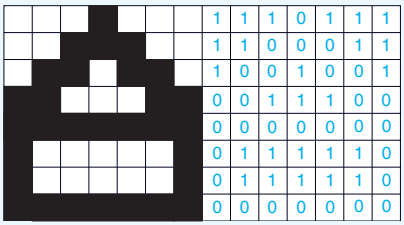
Такім чынам, для апісання адрэзка неабходна 7 лікавых значэнняў, якія апісваюць яго ўласівасці. Гэтых значэнняў дастаткова для апісання адрэзка любога памеру, колеру і таўшчыні.

Круг:

- каардынаты цэнтра круга і яго радыус;
- значэнні шырыні контурнай лініі, колеру контуру, тыпу лініі контуру акружнасці;
- колер заліўкі ўнутранай вобласці, абмежаванай акружнасцю.

Для апісання ўласцівасцей круга можа выкарыстоўвацца 3—7 лікавых значэнняў. Каардынаты цэнтра і радыус з'яўляюцца абавязковымі параметрамі, астатнія параметры могуць адсутнічаць.

Прыклад 14.4. Кадзіраванне чорна-белага відарыса:



1	1	1	0	1	1	1
1	1	0	0	0	1	1
1	0	0	1	0	0	1
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Памер відарыса 7×8 пікселяў, інфармацыйны аб'ём відарыса роўны $8 \cdot 7 \cdot 1$ біт = 56 біт = 7 байт.

Прыклад 14.5. Кадзіраванне аднаго пікселя відарыса.

Колькасць колераў	Колькасць біт для кадзіравання
$2^2 = 4$	2 біты
$2^3 = 8$	3 біты
$2^4 = 16$	4 біты
$2^8 = 256$	8 біт (1 байт)
$2^{16} = 65536$	16 біт (2 байты)
$2^{24} = 16777216$	24 біт (3 байты)

Прыклад 14.6. Кадзіраванне колеру пры выкарыстанні палітры з 16 колераў.

Яркасць	Red	Green	Blue	Колер
0	0	0	0	чорны
0	0	0	1	сіні
0	0	1	0	зялёны
0	0	1	1	блакітны
0	1	0	0	чырвоны
0	1	0	1	фіялетавы
0	1	1	0	карычневы
0	1	1	1	шэры
1	0	0	0	цёмна-шэры
1	0	0	1	ярка-сіні
1	0	1	0	ярка-зялёны
1	0	1	1	ярка-блакітны
1	1	0	0	ярка-чырвоны
1	1	0	1	ярка-ружовы
1	1	1	0	ярка-жоўты
1	1	1	1	белы

матрыцу відарыса з нулёў і адзінак (прыклад 14.4).

Для чорна-белага відарыса інфармацыйны аб'ём пікселя роўны аднаму біту. Адпаведна, інфармацыйны аб'ём відарыса ў бітах будзе роўны колькасці пікселяў у відарысе — здабытку шырыні на даўжыню відарыса.

Чым больш колераў у відарысе, тым больш біт спатрэбіцца для кадзіравання аднаго пункта кропкі (прыклад 14.5).

На экране манітора колер пікселя відарыса фарміруецца змешваннем трох колеравых прамяняў: чырвонага (англ. ***Red***), зялёнага (англ. ***Green***) і сіняга (англ. ***Blue***). Таму пры кадзіраванні колеравых відарысаў выкарыстоўваецца колеравая мадэль RGB. У сучаснай версіі мадэлі RGB на кожны піксель адводзіцца 24 біты, па 8 біт на кожны з трох асноўных колераў, што дае магчымасць закадзіраваць 16,7 млн адценняў.

Калі для кожнага з асноўных колераў выкарыстоўваць меншую колькасць біт, то, адпаведна, можна закадзіраваць і меншую колькасць колеравых адценняў. Кадзіраванне колераў пры выкарыстанні 16-колеравай палітры прыведзена ў прыкладзе 14.6. У гэтым выпадку інфармацыйны аб'ём кожнага пікселя складае 4 біты.

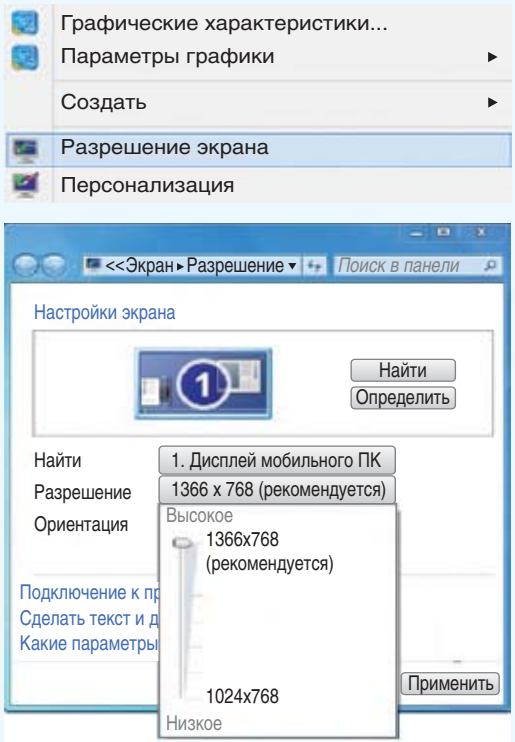
Якасць відарыса на экране залежыць ад разрашальнай здольнасці манітора і глыбіні колеру. Любы графічны відарыс на экране манітора складаецца з радкоў, якія змяшчаюць пэўную колькасць пікселяў. Маніторы могуць мець розныя разрашальныя

здольнасці: 1024×768, 1280×1024, 1366×768, 1920×1080 і інш. Разрашэнне экрана можа быць зменена. Для гэтага ў кантэкставым меню **Рабочага стала** трэба выбраць каманду **Разрешение экрана** (прыклад 14.7).

Глыбіня колеру вызначаецца колькасцю біт, якія выкарыстоўваюцца для кадзіравання колеру пікселя. Сучасныя маніторы падтрымліваюць глыбіню колеру 32 біты: 24 біты захоўваюць код колеру ў RGB-палітры, яшчэ 8 біт адводзяцца на захоўванне значэнняў празрыстасці колеру (альфа-канал).

У файле з графічным растравым відарысам захоўваецца інфармацыя пра колер кожнага пікселя відарыса. У такім выглядзе захоўваюцца відарысы ў фармаце BMP. Іншыя растравыя фарматы (JPEG, GIF, PNG) захоўваюць відарыс у сціснутым выглядзе: пры захаванні ў дачыненні да відарыса, які на экране прадстаўлены матрыцай пікселяў, ужываюць алгарытмы архівіравання. Пры захаванні ў фармаце GIF колькасць колераў памяншаецца да 256. Пры захаванні ў фармаце JPEG захоўваецца інфармацыя не пра кожны піксель, а пра групу пікселяў, пры гэтым частка інфармацыі губляецца. Такое сціскание неабарачальнае, аднавіць відарыс у зыходным выглядзе немагчыма. Аднак чалавечае вока не заўсёды здольна заўважыць змяненні, таму фармат JPEG з'яўляецца адным з самых распаўсюджаных для кампактнага захоўвання фатаграфій. Пры захаванні відарыса ў фармаце PNG

Прыклад 14.7. Змяненне разрашэння экрана.



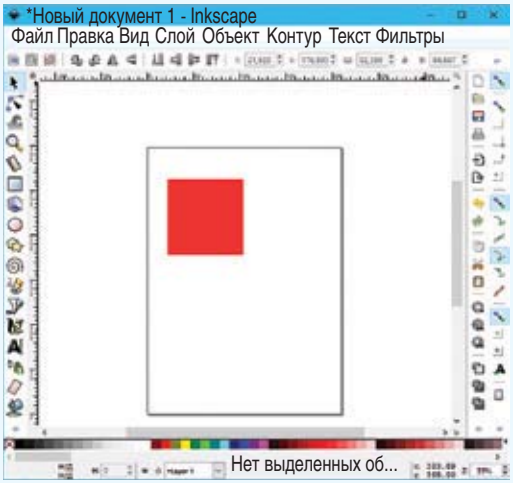
Навуковая дысцыпліна, якая вывучае пытанні вымярэння колеравых характарыстык, называецца *каларыметрыяй* (ці *метралогіяй колеру*).

Навуковую аснову каларыметрыі як спалучэння некалькіх асноўных колераў паклаў Ісаак Ньютан. Ён у 1676 г. з дапамогай трохграннай прызмы расклаў белае сонечнае святло на колеравы спектр і вылучыў сем асноўных колераў: чырвоны, аранжавы, жоўты, зялёны, блакітны, сіні і фіялетавы.

У 1756 г. М. В. Ламаносаў сфармуляваў трохкампанентную тэорыю колеру. Да гэтага лічылася, што колер складаецца з сямі складальнікаў.

Праз стагоддзе Герман Грасман увёў для яе матэматычны апарат.

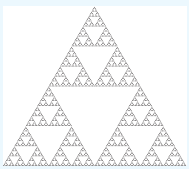
Прыклад 14.8. Малюнак у Inkscape:



Прагляд файла малюнка ў NotePad:

```
<g
   inkscape:label="Layer 1"
   inkscape:groupmode="Layer"
   id="layer1">
   <rect
     style="opacity:1;fill:#ff0000;
     id="rect4485"
     width="50.648811"
     height="50.648811"
     x="70.303574"
     y="109.5238"
     rx="1.984375"
     ry="0.26458332"  />
  </g>
</svg>
```

Прыклад 14.9. Фрактальная графіка.



Слова *фрактал* утворана ад лацінскага *fractus* і ў перакладзе абазначае «які складаецца з фрагментаў». Яно было прапанавана матэматыкам Бенуа Мандэль-Бротам у 1975 г. для абазначэння самападобных структур.

выкарыстоўваецца алгарытм сціскання без страт. Фармат PNG прызначаны першым чынам для выкарыстання ў Інтэрнэце.

У файле з вектарным відарысам захоўваюцца матэматычныя значэнні ўласцівасцей аб'ектаў відарыса, якія неабходны для яго пабудовы. Файлы фармату SVG можна праглядаць і рэдагаваць у тэкставым выглядзе — напрыклад, у рэдактары NotePad (прыклад 14.8).

Фрактальная графіка, як і вектарная, засноўваецца на матэматычных вылічэннях. Базавым элементам фрактальнай графікі з'яўляецца матэматычная **формула**. Гэта прыводзіць да таго, што ў памяці камп'ютара не захоўваецца ніякіх аб'ектаў, а відарыс будуецца па ўраўненнях. Пры дапамозе гэтага спосабу можна будаваць як найпрасцейшыя відарысы, так і складаныя ілюстрацыі, якія імітуюць ландшафты (прыклад 14.9).

14.2. Кадзіраванне гуку

Сучасныя камп'ютарныя ўстройствы абсталяваны ўстройствамі для ўводу і выводу гукавой інфармацыі. Паняцце *гук* цесна звязана з паняццем *хваля*. Як і любая хваля, гук мае амплітуду і частату. Амплітуда характарызуе гучнасць гуку. Частата вызначае тон, вышыню. Звычайны чалавек здольны чуць гукавыя ваганні ў дыяпазоне частот ад 16—20 Гц да 15—20 кгц.

Пры алічбоўванні гук дыскрэтызуецца. Аналага-лічбавы пераўтваральнік, убудаваны ў гукавую карту, выконвае замеры амплітуды гукавой

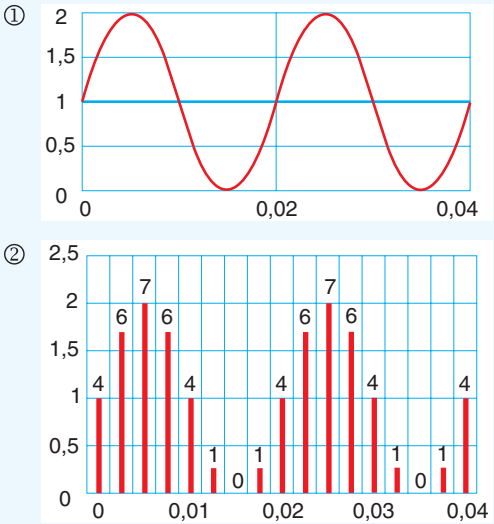
хвалі праз роўныя прамежкі часу. Атрыманыя лікавыя значэнні пераўтвараюцца ў двайковы код і захоўваюцца (прыклад 14.10).

Колькасць вымярэнняў за адну секунду вызначае частату **дыскрэтызацыі** гуку. Дакладнасць пераўтварэння залежыць ад разраднасці АЛП. Разраднасць АЛП характарызуе колькасць дыскрэтных значэнняў, якія пераўтваральнік можа выдаць на выхадзе. Напрыклад, двайковы 8-разрадны АЛП здольны выдаць 256 дыскрэтных значэнняў (0...255), $2^8 = 256$. З разраднасцю звязана **разрашэнне АЛП** — мінімальнае змяненне велічыні аналагавага сігналу, якое можа быць ператворана. Разрашэнне роўна рознасці значэнняў, якія адпавядаюць максімальнаму і мінімальнаму выхадному коду, падзеленай на колькасць выхадных дыскрэтных значэнняў.

Гукавы файл працягласцю ў 1 с пры частаце дыскрэтызацыі 8 КГц і разраднасці 8 біт будзе займаць аб'ём 7,8 Кбайт. Пры павелічэнні частаты дыскрэтызацыі да 44,1 КГц і разраднасці 24 біты аб'ём файла павялічыцца да 129,2 Кбайт.

Каб запісаць стэрэагук, трэба адначасова кадзіраваць два незалежныя каналы гуку. Колькасць каналаў можа быць вялікай: 4 (квадра), 6 (Dolby Digital). Сёння існуюць тэхналогіі, якія падтрымліваюць да 16 гукавых каналаў. Прайграванне шматканальных фанаграм праз сістэму гучнагаварыцеляў, размешчаных па акружнасці ад слухача, называюць **аб'ёмным гукам.**

Прыклад 14.10. Часавая дыяграма кадзіравання гуку: 1 — аналагавы сігнал; 2 — дыскрэтны сігнал.



Для кадзіравання будзем выкарыстоўваць трохразрадны двайковы код. З дапамогай такога кода можна закадзіраваць 8 розных значэнняў. Разаб'ём дыяпазон вымярэння амплітуды сігнала на 8 узроўняў. Кожнаму адліку сігнала прысвоім найбліжэйшы да яго нумар ад 0 да 7. Далей выканаем кадзіраванне атрыманых значэнняў сігнала трохразрадным двайковым кодам (у табліцы прыведзены першыя 6 значэнняў).

Час	Значэнне	Код
0	4	100
0,0025	6	110
0,005	7	111
0,0075	6	110
0,01	4	100
0,0125	1	001
0,015	0	000
0,0175	1	001

Максімальнае і мінімальнае значэнні амплітуды сігнала роўныя 2 і 0 адпаведна. Разрашэнне АЛП у дадзеным выпадку вызначаецца як 2 / 8 = 0,25.

У шматканальным гуку адзін канал выкарыстоўваюць для нізкачастотных эфектаў (выводзіцца на сабвуфер). Паколькі дыяпазон частот гэтага канала вельмі абмежаваны (у параўнанні з іншымі каналамі), то часта яго абазначаюць «.1». Тады абазначэнне 5.1 гаворыць пра тое, што гэта 5 каналаў з поўным дыяпазонам частот і 1 канал для нізкачастотных эфектаў. Агульная колькасць каналаў 6.

Прыклад 14.11. Выкарыстоўваемыя частоты дыскрэтызацыі гуку.

Частата	Устройства
8 000 Гц	Тэлефон, дастаткова для маўлення
22 050 Гц	Радыё
44 100 Гц	Audio CD
48 000 Гц	DAT
96 000 Гц	DVD-Audio (MLP 5.1)
5 644 800 Гц	Прафесійныя ўстройствы запісу

Прыклад 14.12. Гукавыя фарматы.

Фармат	Апісанне
WAV	Waveform audio format — фармат без сціскання даных
FLAC	Free Lossless Audio Codec — сціскание без страт (да 55 % ад зыходнага памеру)
MP3	MPEG-1 Audio Layer 3, сціскание з улікам успрыняцця чалавекам (да 10 % ад зыходнага памеру)
WMA	Windows Media Audio быў прадстаўлены як замена MP3. У апошніх версіях фармату, пачынаючы з Windows Media Audio 9.1, прадугледжана кадзіраванне без страты якасці, шматканальнае кадзіраванне аб'ёмнага гуку і кадзіраванне голасу

Чым вышэйшая частата дыскрэтызацыі і разраднасць, тым больш якасным атрымліваецца гук (прыклад 14.11). Аднак з павелічэннем частаты ўзрастае аб'ём памяці, неабходны для захоўвання лічбавага сігналу, а з павелічэннем разраднасці — і вылічальная нагрузка на лічбавыя пераўтваральнікі.

Каб паменшыць аб'ём, які займаюць лічбавыя аўдыяданыя, ужываюць розныя метады сціскання (прыклад 14.12). Пры сцісканні гуку без страт да зыходнага гуку ўжываюць алгарытмы архівіравання. Магчыма выдаленне залішніх даных — сувязей паміж суседнімі адлікамі лічбавага гукавога сігналу. Сцісканне гуку са стратамі заснавана на недасканаласці чалавечага слыху (чалавек не ўспрымае звышнізкія і звышвысокія частоты, больш слабы сігнал становіцца нячутным на фоне больш моцнага і інш.).

14.3. Кадзіраванне відэа

Відэа захоўваецца на дыску ў выглядзе файлаў, якія змяшчаюць відэа-, аўдыё- і іншыя патокі, а таксама метаданыя. Відэафайл часта называюць медыякантэйнерам. У любы момант з кантэйнера можна выняць, напрыклад, відэа ці аўдыёдарожкі, перакадзіраваць іх і змясціць у іншы кантэйнер, г. зн. змяніць фармат відэафайла. Існуе некалькі фарматаў відэакантэйнераў (прыклад 14.13).

Кадзіраванне гукавога суправаджэння відэаінфармацыі нічым не адрозніваецца ад кадзіравання гуку.

Відарыс у відэа складаецца з асобных кадраў, якія мяняюцца з пэўнай

частатой. Кадр кадзіруецца як звычайны растравы відарыс.

Відэаданыя характарызуюцца частатой кадраў і экранным разрашэннем. Калі частата змены кадраў роўна 25, то для кожнай секунды відэа неабходна захоўваць у памяці 25 кадраў. Разрашэнне для відэа звычайна складае 768×484 (для стандарту NTSC) або 768×576 (для стандартаў PAL і SECAM).

У аснове кадзіравання каляровага відэа ляжыць стандартная мадэль RGB.

Калі ўявіць кожны кадр відарыса як асобны малюнак, то відэавідарыс будзе займаць вельмі вялікі аб'ём. Напрыклад, адна секунда запісу ў сістэме PAL будзе займаць 25 Мбайт. Таму на практыцы выкарыстоўваюцца розныя алгарытмы сціскання для памяншэння аб'ёму відэаданых (прыклад 14.14). Для прагляду такога відэа патрэбен кодэк.

Кодэк (CoDec) — гэта скарачэнне слоў *кампрэсар* і *дэкампрэсар*. Кодэк — гэта набор файлаў, драйвераў і бібліятэк, неабходных для ўпакоўвання відэа ці гукавога файла ў сціснуты фармат і прайгравання сціснутага файла. Кодэк можа адсочваць масівы пунктаў відарыса з аднолькавымі значэннямі (напрыклад, сіні колер мора) і, замест таго каб запамінаць інфармацыю пра кожны пункт (яркасць і колер), запісаць толькі першы (ключавы) пункт і колькасць паўтораў гэтага пункта да моманту змянення яго колеру.

Прыклад 14.13. Фарматы відэафайлаў.

Фармат	Апісанне
AVI (Audio-Video Interleaved)	Уяўляе сабой кантэйнер, які можа змяшчаць струмені чатырох тыпаў — Video, Audio, MIDI, Text
MP4	Сучасны фармат файлаў для захоўвання лічбавых відэа- і аўдыёструменяў, які з'яўляецца часткай стандарту MPEG-4 (гл. прыклад 14.14). Файлы ў гэтым фармаце можна прайграць практычна на любых устройствах, пачынаючы ад смартфонаў і заканчваючы гульнёвымі прыстаўкамі
FLV	Flash Video — медыякантэйнер, які выкарыстоўваецца ў сетцы Інтэрнэт
MOV	Фармат файла, распрацаваны кампаніяй Apple для захоўвання відэа, графікі, анімацыі і 3D

Прыклад 14.14. Стандарты сціскання відэа.

MPEG
(Moving Pictures Expert Group)

Адзін з асноўных стандартаў сціскання. Мае разнавіднасці:

MPEG-1 — фармат сціскання для кампакт-дыскаў (CD-ROM);

MPEG-2 — фармат сціскання для DVD-дыскаў, лічбавага тэлебачання;

MPEG-4 — фармат, які памяншае відэаструмень мацней, чым MPEG-2, але захоўвае добрую якасць.

HD

High Definition — фармат высокага разрашэння і асаблівай выразнасці. Можа выкарыстоўваць разрашэнне 1920×1080.

Windows Media

Распрацаваны кампаніяй Microsoft і прызначаны для захоўвання сціснутага відэа і гуку.

Прыклад 14.15. Відарыс складаецца з $128 \cdot 128 = 2^7 \cdot 2^7 = 2^{14}$ пікселяў. Для захоўвання відарыса вылучана 4 Кбайт $= 4 \cdot 2^{10}$ байт $= 2^{12}$ байт $= 2^{15}$ біт. Значыць, інфармацыйны аб'ём аднаго пікселя роўны $2^{15} / 2^{14} = 2$ біт. З дапамогай 2 біт можна закадзіраваць $2^2 = 4$ колеры.

Прыклад 14.16. Інфармацыйны аб'ём аднаго пікселя роўны $3 \cdot 4 = 12$. Колькасць колераў $2^{12} = 4096$.

Прыклад 14.17. Памер фатаграфіі ў цалях 4×4, паколькі $10 / 2{,}5 = 4$.

Колькасць пікселяў

$$4 \cdot 4 \cdot 400 \cdot 400 = 2^8 \cdot 10000.$$

Інфармацыйны аб'ём

$$2^8 \cdot 10000 \cdot 24 \text{ біт} =$$
$$= 2^8 \cdot 2^4 \cdot 625 \cdot 2^3 \cdot 3 \text{ біт} =$$
$$= 2^{15} \cdot 3 \cdot 5^4 \text{ бит} = 2^{12} \cdot 3 \cdot 5^4 \text{ байт} =$$
$$= 4 \cdot 3 \cdot 5^4 \text{ Кбайт} = 7500 \text{ Кбайт}.$$

Прыклад 14.18. Частата 8 кГц абазначае, што за 1 с было выканана 8000 вымярэнняў. Для захоўвання значэння аднаго вымярэння выкарыстоўваецца 8 біт. Тады для ўсіх вымярэнняў:

$$8 \cdot 8 \cdot 8000 = 2^9 \cdot 1000 \text{ біт} =$$
$$= 2^6 \cdot 1000 \text{ байт} = 64000 \text{ Кбайт} =$$
$$= 62{,}5 \text{ Кбайт}.$$

Прыклад 14.19. Інфармацыйны аб'ём стэрэааўдыяфайла можна вылічыць па формуле: $V = 2 \cdot R \cdot t \cdot N$, дзе V — аб'ём аўдыяфайла, R — разраднасць аўдыяадаптара, N — частата дыскрэтызацыі, t — час гучання, множанне на 2 паказвае, што кадзіруюцца два каналы. Тады $t = \dfrac{V}{2RN}$. Пераўтворым зыходныя даныя:

$V = 70$ Мбайт $= 70 \cdot 220 \cdot 8 = 70 \cdot 223$ біт;

$N = 32$ кГц $= 32000$ Гц $= 25 \cdot 1000$.

Тады $t = \dfrac{70 \cdot 2^{23}}{2 \cdot 16 \cdot 2^5 \cdot 1000} = \dfrac{70 \cdot 2^{23}}{2^{10} \cdot (2 \cdot 5)^3} =$

$= \dfrac{70 \cdot 2^{10}}{5^3} = 573{,}44$ с $= 9{,}56$ мін.

Прыклад 14.20. Знойдзем аб'ём графікі: $800 \cdot 600 \cdot 24$ біт $= 11520000$ біт $\approx$ $\approx 1{,}38$ Мбайт. Памер відэа: $1{,}38 \cdot 24 \times$ $\times (5 \cdot 60) = 9936$ Мбайт. Разраднасць пры кадзіраванні гуку роўна 8, паколькі $256 = 2^8$. Памер гуку: $11250 \cdot 8 \cdot 2 \cdot (5 \cdot 60) =$ $= 54000000$ біт $\approx 6{,}4$ Мбайт. Аб'ём відэафайла: $9936 + 6{,}4 = 9942{,}4$ Мбайт $\approx$ $\approx 9{,}7$ Гігабайт.

14.4. Рашэнне задач на кадзіраванне графікі, гуку і відэа

Прыклад 14.15. Для захоўвання відарыса памерам 128×128 пунктаў вылучана 4 Кбайт памяці. Вызначыце, якая максімальная колькасць колераў у палітры.

Прыклад 14.16. Колер пікселя, які фарміруецца прынтарам, вызначаецца трыма складальнікамі: блакітнай, пурпурнай і жоўтай фарбамі. Пад кожны складальнік аднаго пікселя адвялі па 4 біты. У якую колькасць колераў можна размаляваць піксель?

Прыклад 14.17. Фатаграфія памерам 10×10 см была адсканіравана з разрашэннем 400 dpi пры глыбіні колеру 24 біты. Вызначыце інфармацыйны аб'ём атрыманага растравага файла ў кілабайтах (прыняць 1 цалю $=$ $= 2{,}5$ см).

Прыклад 14.18. Вызначыць інфармацыйны аб'ём у Кбайтах монааўдыяфайла працягласцю гучання 8 с пры глыбіні гуку 8 біт і частаце 8 кГц.

Прыклад 14.19. Разлічыць час гучання стэрэааўдыяфайла, які быў закадзіраваны з частатой дыскрэтызацыі 32 кГц. Разраднасць аўдыёадаптара — 16 біт, інфармацыйны аб'ём файла роўны 70 Мбайт.

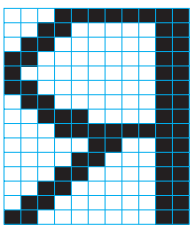
Прыклад 14.20. Які аб'ём будзе мець відэа, якое перадаецца з разрашэннем кадра 800×600 пікселяў з 24-бітавай глыбінёй колеру, скорасцю паказу 24 кадры ў секунду і працягласцю 5 мін? Вядома, што стэрэагук, накладзены на відэа, мае 256 узроўняў гучнасці, частата дыскрэтызацыі роўна 11 250 Гц.

1. Якія два прынцыпы ўяўлення графічных відарысаў выкарыстоўваюцца ў камп'ютарнай графіцы?
2. З чаго складаецца растравы відарыс?
3. Што ўяўляе сабой вектарны відарыс?
4. Чым адрозніваецца растравы відарыс ад вектарнага?
5. Графічныя відарысы якога тыпу выводзяцца на экран манітора?
6. Што разумеюць пад разрашальнай здольнасцю экрана манітора і глыбінёй колеру?
7. Як захоўваюцца растравыя і вектарныя відарсы ў файле?
8. Чым вызначаецца частата дыскрэтызацыі гуку?
9. Што такое разраднасць аналага-лічбавага пераўтваральніка?
10. Як кадзіруецца відэа?

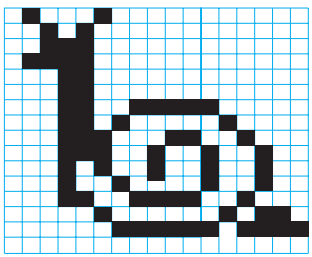
Практыкаванні

1 Стварыце матрыцу з нулёў і адзінак для кадзіравання наступных чорна-белых відарысаў (можна выкарыстоўваць электронныя табліцы; для праверкі правільнасці, падлічыце сумы па радках і слупках). Вызначыце інфармацыйны аб'ём відарысаў.

1.

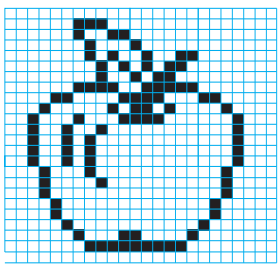


2.

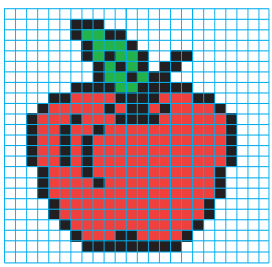


2 Вызначыце інфармацыйны аб'ём змешчаных ніжэй растравых відарысаў (адна клетка — адзін піксель).

1. Чорна-белы відарыс



2. Чатырохколерны відарыс

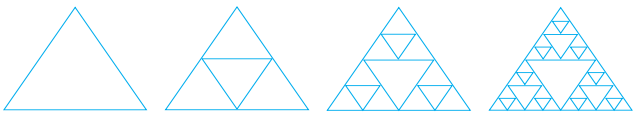


3 16-колерны малюнак змяшчае 500 байт інфармацыі. З колькіх пунктаў ён складаецца?

4 Вызначыце патрэбны аб'ём (у мегабайтах) відэапамяці для рэалізацыі графічнага рэжыму манітора з разрашальнай здольнасцю 1024×768 пікселяў пры колькасці колераў, якія адлюстроўваюцца, 65 536.

5* На малюнку ў прыкладзе 14.9 адлюстравана фігура, якая называецца «трохвугольнік Сярпінскага». Для яе пабудовы можна выкарыстаць наступны алгарытм:

1. Будуем роўнастаронні трохвугольнік (узровень 0).
2. Злучаем сярэдзіны старон пабудаванага трохвугольніка адрэзкамі. Атрымліваецца 4 новыя трохвугольнікі. З зыходнага трохвугольніка выдаляецца ўнутранасць сярэдняга трохвугольніка. Атрымліваем 3 трохвугольнікі ўзроўню 1.
3. Робячы тое ж самае з кожным з трохвугольнікаў першага ўзроўню, атрымаем мноства, якое складаецца з 9 роўнастаронніх трохвугольнікаў другога ўзроўню.
4. Паўтараем працэс да патрэбнага ўзроўню.



Рэалізуйце дадзены алгарытм у асяроддзі праграміравання.

6 Вызначыце працягласць гучання стэрэааўдыяфайла, які займае 468,75 Кбайт памяці пры глыбіні гуку 16 біт і частаце 48 кГц.

7 Пры пераводзе ў дыскрэтную форму аналагавага сігналу працягласцю 2 мін 8 с выкарыстоўвалася частата дыскрэтызацыі 32 Гц і 16 узроўняў дыскрэтызацыі. Знайсці ў байтах памер атрыманага кода аналагавага сігналу.

8 Эксперыментальна было вызначана, што на часавым адрэзку [0; 20] ампліту-да гукавога сігналу змянялася ў адпаведнасці з законам $A(t) = 2\ \sin 1{,}7x \cdot \sin 0{,}2x$. Выкарыстоўваючы электронныя табліцы, пабудуйце часавыя дыяграмы кадзіравання гуку.

1. Дыяграму аналагавага сігналу (у Excel можна выкарыстоўваць тып дыяграмы — кропкавая).
2. Дыяграму дыскрэтнага сігналу (у Excel можна выкарыстоўваць тып дыяграмы — гістаграма).

9 Які аб'ём будзе мець чорна-белае відэа, якое перадаецца з разрашэннем кадра 800×600, скорасцю паказу 24 кадры ў секунду і працягласцю 30 мін без гуку?

10 Кадры відэазапісу закадзіраваны ў рэжыме сапраўднага колеру (24 біты на піксель) і змяняюцца з частатой 25 кадраў у секунду. Кадр мае памеры 720×480 пікселяў. Частата дыскрэтызацыі 22 кГц, глыбіня кадзіравання гуку 16 біт. Ацаніце аб'ём мінуты відэазапісу ў мегабайтах (з дакладнасцю да дзясятых), калі файл запісаны з 10-кратнай ступенню сціснутасці.

11 Камера здымае відэа без гуку з частатой 60 кадраў у секунду, пры гэтым відарысы выкарыстоўваюць палітру, якая змяшчае 224 колераў. Пры запісе файла на сервер атрыманае відэа пераўтвараюць так, што частата кадраў памяншаецца да 20, а відарысы пераўтвараюць у фармат, які выкарыстоўвае палітру з 256 колераў. Іншыя пераўтварэнні і іншыя метады сціскання не выкарыстоўваюцца. 10 секунд ператворанага відэа ў сярэднім займаюць 512 Кбайт. Колькі Мбайт у сярэднім займае 1 мінута зыходнага відэа?

§ 15. Розныя падыходы да вымярэння інфармацыі

15.1. Змястоўны падыход

Сёння інфармацыя з'яўляецца адным з асноўных рэсурсаў чалавецтва. Таму важнымі з'яўляюцца адказы на пытанні: як шмат інфармацыі мы атрымалі, перадалі, апрацавалі, стварылі.

Пры фізічных вымярэннях велічыню параўноўваюць з эталонам. А з чым параўноўваць інфармацыю?

Вядома некалькі падыходаў да вымярэння колькасці інфармацыі.

Пры **змястоўным падыходзе** вымярэнне інфармацыі адбываецца з пункту гледжання яе зместу, г. зн. вызначаецца, у якой меры атрыманая інфармацыя (веды) памяншае няведанне. Чалавек атрымлівае веды пры дапамозе паведамленняў. Чым больш папаўняе нашы веды паведамленне, тым большая колькасць інфармацыі ў ім знаходзіцца (прыклад 15.1).

Заснавальнікам такога падыходу да вымярэння інфармацыі з'яўляецца К. Шэнан, які ўвёў прыведзенае ніжэй азначэнне.

Паведамленне, якое памяншае няпэўнасць ведаў у два разы, нясе **1 біт** інфармацыі.

Няпэўнасць ведаў пра вынік некаторай падзеі — колькасць магчымых вынікаў.

Калі ў некаторым паведамленні змяшчаюцца звесткі пра тое, што адбылася адна з N роўнаімаверных падзей, то колькасць інфармацыі i, якая змяшчаецца ў паведамленні, можна вызначыць з формулы Хартлі: $N = 2^i$ (прыклад 15.2).

Клод Элвуд Шэнан (1916—2001) — амерыканскі інжынер, крыптааналітык і матэматык. З'яўляецца заснавальнікам тэорыі інфармацыі.

Прыклад 15.1. У вас сёння кантрольная па матэматыцы. Настаўнік звычайна дае 2 варыянты заданняў. Да кантрольнай вы не ведаеце свой варыянт, таму няпэўнасць ведаў роўна 2. Калі варыянтаў на кантрольнай 4, то няпэўнасць ведаў роўна 4.

Ральф Вінтон Лаян Хартлі (1888—1970) — амерыканскі вучоны-электроншчык. Прапанаваў генератар Хартлі, пераўтварэнне Хартлі і зрабіў уклад у тэорыю інфармацыі, увёўшы ў 1928 г. лагарыфмічную меру інфармацыі: $i = \log_2 N$.

Прыклад 15.2. Колькасць інфармацыі, якую вы атрымаеце, даведаўшыся пра свой варыянт кантрольнай работы, можна разлічыць па формуле Хартлі.

Калі варыянтаў два, то $2 = 2^i$, значыць, $i = 1$. Вы атрымаеце 1 біт інфармацыі.

Калі варыянтаў 4, то $4 = 2^i$, значыць, $i = 2$. Вы атрымаеце 2 біты інфармацыі.

Калі варыянтаў 6, то $6 = 2^i$, значыць, $i \approx 2{,}58$. Вы атрымаеце 2,58 біта інфармацыі. Для атрымання значэння i у гэтым выпадку трэба палічыць значэнне $i = \log_2 6$ (напрыклад, на калькулятары або па табліцы[1]).

Прыклад 15.3. Пры камп'ютарным наборы тэксту на рускай мове звычайна выкарыстоўваецца 32 літары (літара «ё» ўжываецца вельмі рэдка). Тады, згодна з формулай Хартлі, $32 = 2^5$, адна літара рускага алфавіта нясе 5 біт інфармацыі.

Прыклад 15.4. Адзінкі вымярэння аб'ёмаў інфармацыі.

Кілабайт (Кбайт):
2^{10} = 1024 байты

Мегабайт (Мбайт):
2^{20} = 1024 кілабайты = 1 048 576 байт

Гігабайт (Гбайт):
2^{30} = 1024 мегабайты = 1 073 741 824 байты

Тэрабайт (Тбайт):
2^{40} = 1024 гігабайты = 1 099 511 627 776 байт

Петабайт (Пбайт):
2^{50} = 1024 тэрабайты =
= 1 125 899 906 842 624 байты

Эксабайт (Эбайт):
2^{60} = 1024 петабайты =
= 1 152 921 504 606 846 976 байт

Зэтабайт (Збайт):
2^{70} = 1024 эксабайты =
= 1 180 591 620 717 411 303 424 байты

Ётабайт (Йбайт):
2^{80} = 1024 зэтабайты =
= 1208925819614629174706176 байт

З пункту гледжання тэорыі вымярэнняў адзінкі вымярэння колькасці інфармацыі, у назве якіх ёсць часткі «кіла», «мега» і інш., некарэктныя. Гэтыя прыстаўкі выкарыстоўваюцца ў метрычнай сістэме мер, у якой у якасці множнікаў кратных адзінак ужываецца каэфіцыент 10^n, дзе n = 3, 6, 9 і г. д.

15.2. Алфавітны падыход

Калі чалавек атрымлівае тэкставае паведамленне, то колькасць інфармацыі можа быць вымерана колькасцю сімвалаў у ім. Аднак кожны сімвал алфавіта таксама нясе якую-небудзь колькасць інфармацыі. Калі выказаць меркаванне, што ўсе сімвалы алфавіта сустракаюцца ў тэксце з аднолькавай частатой (роўнаімаверна), то колькасць інфармацыі i, якую нясе кожны сімвал, вылічваецца па формуле Хартлі: $N = 2^i$, дзе N — магутнасць алфавіта (прыклад 15.3). Пад магутнасцю алфавіта разумеюць колькасць сімвалаў у ім.

Алфавітны (аб'ёмны) падыход выкарыстоўваецца, калі для пераўтварэння, захоўвання і перадачы інфармацыі ўжываюць тэхнічныя сродкі.

Пры выкарыстанні двайковага алфавіта адзін сімвал нясе 1 адзінку інфармацыі — **1 біт.**

Для вымярэння аб'ёмаў інфармацыі ўжываюць вытворныя адзінкі вымярэння (прыклад 15.4).

Для двайковага ўяўлення тэкстаў у камп'ютары часта выкарыстоўваецца васьміразрадны код. З яго дапамогай можна закадзіраваць алфавіт з 256 сімвалаў. Адзін сімвал з алфавіта магутнасцю $256 = 2^8$ нясе ў тэксце 8 біт інфармацыі. Такая колькасць інфармацыі называецца **байтам**.

Аб'ём тэксту вымяраецца ў байтах. Пры васьміразрадным кадзіраванні 1 сімвал = 1 байт, і інфармацыйны аб'ём тэксту вызначаецца колькасцю сімвалаў у ім. Калі ўвесь тэкст скла-

[1] Табліца двайковых лагарыфмаў: http://sokolova-aa.ru/cribs/tablitsa-dvoichnykh-logarifmov-tselykh-chisel-ot-1-do-64 (дата доступу: 28.07.2019).

даецца з K сімвалаў, то пры алфавітным падыходзе аб'ём V наяўнай у ім інфармацыі роўны: $V = K \cdot i$, дзе i — інфармацыйная вага аднаго сімвала ў алфавіце, які выкарыстоўваецца.

15.3. Імавернасны падыход

У жыцці розныя падзеі адбываюцца з рознай імавернасцю. Падзея «летам ідзе снег» малаімаверная, а ў падзеі «восенню ідзе дождж» імавернасць вялікая. Калі ў скрынцы 10 чырвоных шароў і 40 зялёных, то імавернасць дастаць не гледзячы зялёны шар большая, чым імавернасць дастаць чырвоны.

Для колькаснага вымярэння імавернасці выкарыстоўваюць наступны падыход: калі агульная колькасць магчымых зыходаў якой-небудзь падзеі роўна N, а K з іх — тыя, у якіх мы зацікаўлены, то імавернасць падзеі, што нас цікавіць, можа быць палічана па формуле $p = \frac{K}{N}$ (прыклад 15.5).

Чым меншая імавернасць падзеі, тым больш інфармацыі змяшчае паведамленне пра тое, што гэта падзея адбылася.

Імавернасны падыход ужываецца для вымярэння колькасці інфармацыі пры наступленні падзей, якія маюць розную імавернасць. Сувязь паміж імавернасцю падзеі і колькасцю інфармацыі ў паведамленні пра яе адлюстроўваецца формулой $\frac{1}{p} = 2^i$, дзе p — імавернасць падзеі, а i — колькасць інфармацыі (прыклад 15.6).

У 1999 г. зацвердзілі шэраг новых прыставак для адзінак вымярэння колькасці інфармацыі: кібі (kibi), мебі (mebi), гібі (gibi), тэбі (tebi), пеці (peti), эксбі (exbi)[1]. У цяперашні час выкарыстоўваюцца і тыя і іншыя прыстаўкі.

Прыклад 15.5. У каробцы 16 чырвоных шароў і 48 зялёных. Якая імавернасць дастаць зялёны шар не гледзячы? Чырвоны?

Усяго ў каробцы $N = 16 + 48 = 64$ шары. Нас цікавіць зялёны шар. Удачны выпадак — дастаць любы з 48 зялёных шароў. Таму $p = \frac{48}{64} = 0{,}75$. Аналагічна атрымаем імавернасць дастаць чырвоны шар: $p = \frac{16}{64} = 0{,}25$. Значыць, імавернасць выцягнуць зялёны шар у 3 разы большая, чым выцягнуць чырвоны шар.

* Калі адбылося некалькі рознаімавернасных падзей, то колькасць інфармацыі можна вызначаць па формуле Шэнана, прапанаванай ім у 1948 г.: $I = -(p_1 \log_2 p_1 + p_2 \log_2 p_2 + \ldots\ p_N \log_2 p_N)$, дзе I — колькасць інфармацыі; N — колькасць магчымых падзей; p_i — імавернасць i-й падзеі.

Лёгка заўважыць, што калі імавернасці $p_1, \ldots, p_N$ роўныя паміж сабой, то кожная з іх роўна $1/N$ і формула Шэнана пераўтвараецца ў формулу Хартлі.

Прыклад 15.6. Якую колькасць інфармацыі нясе паведамленне «З каробкі дасталі чырвоны шар» для прыкладу 15.5?

Па формуле $\frac{1}{p} = 2^i$ атрымліваем $\frac{1}{0{,}25} = 2^i \Rightarrow 4 = 2^i$. Тады $i = 2$, г. зн. мы атрымалі 2 біты інфармацыі.

[1] https://ru.wikipedia.org/wiki/Двоичные_приставки (дата доступу: 28.07.2019).

Прыклад 15.7. Выпадзенне кожнай грані кубіка роўнаімавернае. Таму колькасць інфармацыі ад аднаго выніку кідання знаходзіцца з ураўнення $2^i = 6$. Тады $2^i = 6 < 8 = 2^3$, $i = 3$ біты.

Можна разважаць і так:

$i = \log_2 6 = 2{,}585$ біты ≈ 3 біты.

Прыклад 15.8. 11 Кбайт $= 11 \cdot 1024 =$ $= 11\,264$ байты. Паколькі колькасць байт роўна колькасці сімвалаў, то выкарыстана васьмібітная кодавая табліца. Магутнасць алфавіта: $2^8 = 256$.

Прыклад 15.9. Трэба закадзіраваць 65 роўнаімавернасных значэнняў. Па формуле Харлі: $2^i = 65 < 128 = 2^7$, $i = 7$ біт.

Прыклад 15.10. Няхай у скрыні x жоўтых мячоў. Тады імавернасць дастаць жоўты мяч роўна $\frac{x}{32}$. Падстаўляем у формулу, якая звязвае імавернасць з колькасцю інфармацыі:

$$\frac{1}{p} = 2^i \Rightarrow \frac{1}{x/32} = 2^4 \Rightarrow \frac{32}{x} = 16 \Rightarrow x = 2.$$

У скрыні 2 жоўтыя мячы.

Прыклад 15.11*. Усяго ў скрыні $10 + 8 + 6 = 24$ кубікі. Імавернасці даставання кубікаў: $p_ч = \frac{10}{24}$, $p_{зял} = \frac{8}{24}$, $p_ж = \frac{6}{24}$. Колькасць інфармацыі па формуле Шэнана:

$$I = -(p_ч \log_2 p_ч + p_{зял} \log_2 p_{зял} + p_ж \log_2 p_ж) =$$
$$= \left(\frac{10}{24}\log_2\frac{10}{24} + \frac{8}{24}\log_2\frac{8}{24} + \frac{6}{24}\log_2\frac{6}{24}\right) \approx$$
$$\approx -(0{,}42 \cdot (-1{,}26) + 0{,}33 \cdot (-1{,}58) +$$
$$+ 0{,}25 \cdot (-2)) = 1{,}5506.$$

15.4. Рашэнне задач на вызначэнне аб'ёму інфармацыі

Прыклад 15.7. Пры гульні ў косткі выкарыстоўваецца кубік з шасцю гранямі. Колькі біт інфармацыі атрымлівае гулец пры кожным кіданні кубіка? Адказ акругліць у большы бок да бліжэйшай цэлай колькасці біт.

Прыклад 15.8. Аб'ём паведамлення роўны 11 Кбайт. Паведамленне змяшчае 11 264 сімвалы. Якая магутнасць алфавіта?

Прыклад 15.9. Вымяраецца тэмпература паветра, якая можа быць цэлым лікам ад –30 да 34 градусаў. Якая найменшая цэлая колькасць біт неабходна, каб закадзіраваць адно вымеранае значэнне?

Прыклад 15.10. У скрыні знаходзяцца 32 тэніснyя мячы, сярод якіх ёсць мячы жоўтага колеру. Наўгад вымаецца адзін мяч. Паведамленне «Выняты мяч жоўтага колеру» нясе 4 біты інфармацыі. Колькі жоўтых мячоў у скрыні?

Прыклад 15.11*. У скрынцы ляжаць кубікі. Вядома, што сярод іх 10 чырвоных, 8 зялёных, 6 жоўтых. Вылічыце імавернасць даставання кубіка кожнага колеру. Колькі інфармацыі нясе паведамленне, што дасталі кубік любога колеру? Для рашэння задачы выкарыстайце формулай Шэнана.

? **1.** У чым сутнасць змястоўнага падыходу да вымярэння інфармацыі?
2. Што абазначае 1 біт інфармацыі пры алфавітным падыходзе да вымярэння інфармацыі?
3. Калі ўжываюць імавернасны падыход да вымярэння інфармацыі?

Практыкаванні

1 Колькі розных гукавых сігналаў можна закадзіраваць з дапамогай 6 біт?

2 Якую колькасць інфармацыі нясе паведамленне пра тое, што чалавек жыве ў першым або другім пад'ездзе, калі ў доме 16 пад'ездаў?

3 Паведамленне пра тое, што ваш сябар жыве на 10-м паверсе, нясе 4 біты інфармацыі. Колькі паверхаў у доме?

4 Азбука Морзэ дазваляе кадзіраваць сімвалы для радыёсувязі, задаючы камбінацыю кропак і працяжнікаў. Колькі розных сімвалаў (лічбаў, літар, знакаў пунктуацыі і г. д.) можна закадзіраваць, выкарыстоўваючы код Морзэ даўжынёй не менш за пяць і не больш за шэсць сігналаў (кропак і працяжнікаў)?

5 У скрыні знаходзіцца 32 тэнісныя мячы, сярод якіх ёсць мячы чорнага колеру. Наўгад вымаецца адзін мяч. Паведамленне «Выняты мяч НЕ чорнага колеру» нясе 3 біты інфармацыі. Колькі чорных мячоў у скрыні?

6 Да свята надзімалі белыя і сінія шарыкі. Белых шарыкаў 24. Паведамленне пра тое, што лопнуў сіні шарык, нясе 2 біты інфармацыі. Колькі ўсяго надзьмулі шарыкаў?

7 У школьнай бібліятэцы 32 стэлажы з кнігамі, на кожным — па 8 паліц. Пецю паведамілі, што патрэбны падручнік знаходзіцца на 2-й паліцы 4-га стэлажа. Якую колькасць інфармацыі атрымаў Пеця?

8 Для рэгістрацыі на некаторым сайце карыстальніку трэба прыдумаць пароль, які складаецца з 10 сімвалаў. У якасці сімвалаў можна выкарыстоўваць дзесятковыя лічбы і шэсць першых літар лацінскага алфавіта, прычым літары выкарыстоўваюцца толькі вялікія. Паролі кадзіруюцца пасімвальна. Усе сімвалы кадзіруюцца аднолькавай і мінімальна магчымай колькасцю біт. Для захоўвання звестак пра кожнага карыстальніка ў сістэме адведзена аднолькавая і мінімальна магчымая цэлая колькасць байт. Які аб'ём будзе займаць інфармацыя пра паролі 1000 карыстальнікаў?

9* У некаторай краіне аўтамабільны нумар даўжынёй 6 сімвалаў складаюць з вялікіх літар (задзейнічана 30 розных літар) і дзесятковых лічбаў у любым парадку. Кожны такі нумар у камп'ютарнай праграме запісваецца мінімальна магчымай і аднолькавай цэлай колькасцю байт (пры гэтым выкарыстоўваюць пасімвальнае кадзіраванне і ўсе сімвалы кадзіруюцца аднолькавай і мінімальна магчымай колькасцю біт). Вызначыце аб'ём памяці ў байтах, які адводзіцца гэтай праграмай для запісу 50 нумароў.

10 У возеры жывуць 12500 акунёў, 25000 печкуроў, а карасёў і шчупакоў па 6250. Якую колькасць інфармацыі нясе паведамленне пра тое, што злавілі печкура? Колькі інфармацыі мы атрымаем, калі зловім якую-небудзь рыбу?

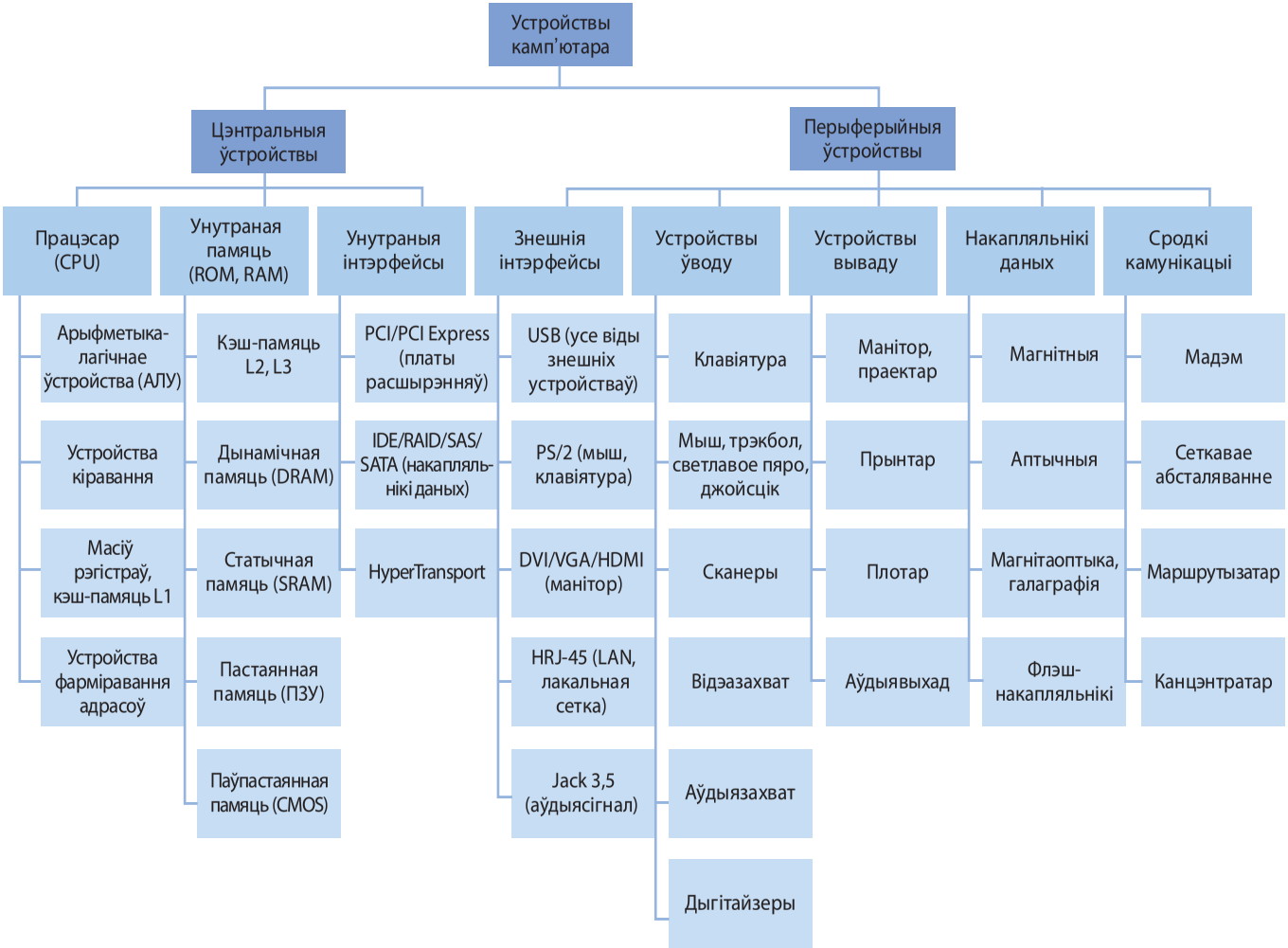
11* Якое паведамленне змяшчае большую колькасць інфармацыі?

1. Бабуля спякла 16 піражкоў. Лера з'ела адзін піражок.
2. Бабуля спякла 12 піражкоў з капустай, 12 піражкоў з павідлам. Маша з'ела адзін піражок.
3. Бабуля спякла 16 піражкоў з капустай, 24 піражкі з павідлам. Міша з'еў адзін піражок.

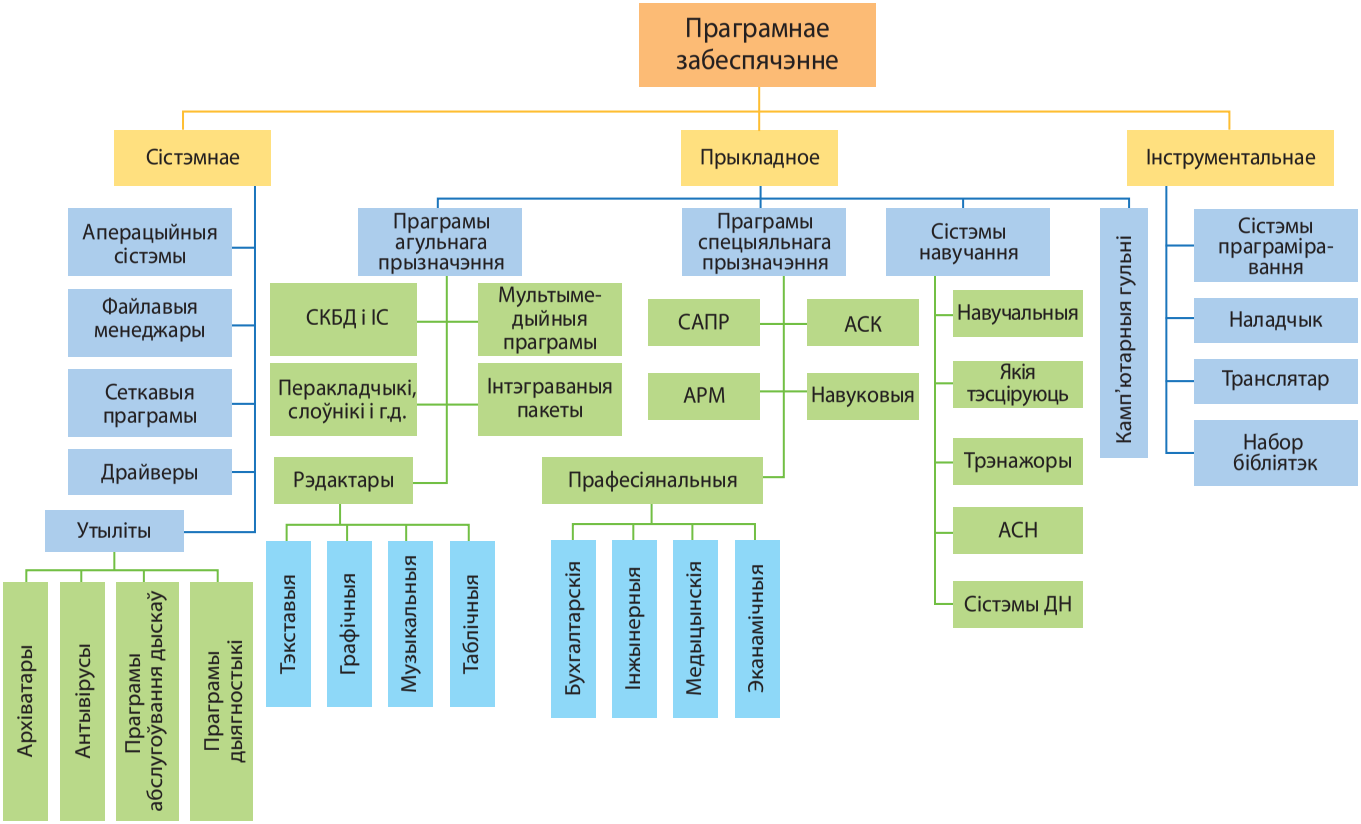
ДАДАТАК ДА ГЛАВЫ 2

Структурная схема камп'ютара

Асноўныя ўстройствы камп’ютара



Класіфікацыя праграмнага забеспячэння паводле прызначэння



СКБД і ІС — сістэмы кіравання базамі даных і інфармацыйныя сістэмы
САПР — сістэмы аўтаматызаванага праектавання
АРМ — аўтаматызаваныя рабочыя месцы
АСК — аўтаматызаваныя сістэмы кіравання
АСН — аўтаматызаваныя сістэмы навучання
ДН — дыстанцыйнае навучанне

Структура кода ў кадзіроўцы UTF-8

Колькасць байт	Значных біт	Першы байт	Шаблон цалкам
1	7	0xxxxxxx	0xxxxxxx
2	11	110xxxxx	110xxxxx 10xxxxxx
3	16	1110xxxx	1110xxxx 10xxxxxx 10xxxxxx
4	21	11110xxx	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Табліца двайковых лагарыфмаў

№	x	№	x	№	x	№	x
1	0,00000	17	4,08746	33	5,04439	49	5,61471
2	1,00000	18	4,16993	34	5,08746	50	5,64386
3	1,58496	19	4,24793	35	5,12928	51	5,67243
4	2,00000	20	4,32193	36	5,16993	52	5,70044
5	2,32193	21	4,39232	37	5,20945	53	5,72792
6	2,58496	22	4,45943	38	5,24793	54	5,75489
7	2,80735	23	4,52356	39	5,28540	55	5,78136
8	3,00000	24	4,58496	40	5,32193	56	5,80735
9	3,16993	25	4,64386	41	5,35755	57	5,83289
10	3,32193	26	4,70044	42	5,39232	58	5,85798
11	3,45943	27	4,75489	43	5,42626	59	5,88264
12	3,58496	28	4,80735	44	5,45943	60	5,90689
13	3,70044	29	4,85798	45	5,49185	61	5,93074
14	3,80735	30	4,90689	46	5,52356	62	5,95420
15	3,90689	31	4,95420	47	5,55459	63	5,97728
16	4,00000	32	5,00000	48	5,58496	64	6,00000

Кодавая табліца сімвалаў стандарта ASCII

№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал
0		16	▶	32		48	0	64	@	80	P	96	`	112	p
1		17	◀	33	!	49	1	65	A	81	Q	97	a	113	q
2		18	↕	34	"	50	2	66	B	82	R	98	b	114	r
3	©	19	‼	35	#	51	3	67	C	83	S	99	c	115	s
4	♥	20	¶	36	$	52	4	68	D	84	T	100	d	116	t
5	♦	21	§	37	%	53	5	69	E	85	U	101	e	117	u
6	♣	22	■	38	&	54	6	70	F	86	V	102	f	118	v
7	•	23		39	'	55	7	71	G	87	W	103	g	119	w
8		24	↑	40	(	56	8	72	H	88	X	104	h	120	x
9		25	↓	41	)	57	9	73	I	89	Y	105	i	121	y
10		26	→	42	*	58	:	74	J	90	Z	106	j	122	z
11	♂	27	←	43	+	59	;	75	K	91	[	107	k	123	{
12	♀	28	∟	44	,	60	<	76	L	92	\	108	l	124	|
13		29	↔	45	-	61	=	77	M	93	]	109	m	125	}
14	♫	30	▲	46	.	62	>	78	N	94	^	110	n	126	~
15		31	▼	47	/	63	?	79	O	95	_	111	o	127	⌂

№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал	№	Сімвал
128	А	144	Р	160	а	176	▒	192	└	208	╨	224	р	240	Ё
129	Б	145	С	161	б	177	▒	193	┴	209	╤	225	с	241	ё
130	В	146	Т	162	в	178	▓	194	┬	210	╥	226	т	242	Є
131	Г	147	У	163	г	179	│	195	├	211	╙	227	у	243	є
132	Д	148	Ф	164	д	180	┤	196	─	212	╘	228	ф	244	Ї
133	Е	149	Х	165	е	181	╡	197	┼	213	╒	229	х	245	ї
134	Ж	150	Ц	166	ж	182	╢	198	╞	214	╓	230	ц	246	Ў
135	З	151	Ч	167	з	183	╖	199	╟	215	╫	231	ч	247	ў
135	И	152	Ш	168	и	184	╕	200	╚	216	╪	232	ш	248	°
137	Й	153	Щ	169	й	185	╣	201	╔	217	┘	233	щ	249	·
138	К	154	Ъ	170	к	186	║	202	╩	218	┌	234	ъ	250	·
139	Л	155	Ы	171	л	187	╗	203	╦	219	█	235	ы	251	√
140	М	156	Ь	172	м	188	╝	204	╠	220	▄	236	ь	252	№
141	Н	157	Э	173	н	189	╜	205	═	221	▌	237	э	253	¤
142	О	158	Ю	174	о	190	╛	206	╬	222	▐	238	ю	254	■
143	П	159	Я	175	п	191	┐	207	╧	223	▀	239	я	255	

(Назва ўстановы адукацыі)

Вучэбны год	Імя і прозвішча навучэнца	Стан вучэбнага дапаможніка пры атрыманні	Адзнака навучэнцу за карыстанне вучэбным дапаможнікам
20 /			
20 /			
20 /			
20 /			
20 /			
20 /			

Вучэбнае выданне

Котаў Уладзімір Міхайлавіч
Лапо Анжаліка Іванаўна
Быкадораў Юрый Аляксандравіч
Вайцеховіч Алена Мікалаеўна

ІНФАРМАТЫКА

Вучэбны дапаможнік для 10 класа
ўстаноў агульнай сярэдняй адукацыі
з беларускай мовай навучання (з электроннымі дадаткамі)

Заг. рэдакцыі *Г. А. Бабаева*. Рэдактар *К. І. Чэрнікава*.
Мастацкі рэдактар *В. М. Карповіч*. Мастак *А. М. Багушэвіч*.
Тэхнічнае рэдагаванне і камп'ютарная вёрстка *І. І. Дуброўскай*.
Карэктары *Г. В. Алешка, А. П. Тхір*.

Падпісана да друку 17.07.2020. Фармат $70 \times 90\,^{1}/_{16}$. Папера афсетная. Гарнітура школьная. Друк афсетны. Умоўн. друк. арк. 8,78. Ул.-выд. арк. 8,0 + 20,0 эл. дадат. Тыраж 11 500 экз. Заказ .

Выдавецкае рэспубліканскае ўнітарнае прадпрыемства «Народная асвета» Міністэрства інфармацыі Рэспублікі Беларусь. Пасведчанне аб дзяржаўнай рэгістрацыі выдаўца, вытворцы, распаўсюджвальніка друкаваных выданняў № 1/2 ад 08.07.2013. Пр. Пераможцаў, 11, 220004, Мінск, Рэспубліка Беларусь.

Адкрытае акцыянернае таварыства «Паліграфкамбінат імя Я. Коласа». Пасведчанне аб дзяржаўнай рэгістрацыі выдаўца, вытворцы, распаўсюджвальніка друкаваных выданняў № 2/3 ад 10.09.2018. Вул. Каржанеўскага, 20, 220024, Мінск, Рэспубліка Беларусь.