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 выходным 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 ($-200 \leqslant X[i], Y[i] \leqslant 200$). 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.

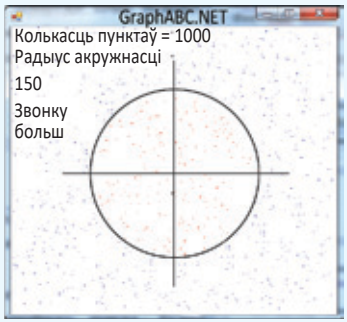
4. Вывад выніку.

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