Funkcijos

Informacinės technologijos

10 - 11 pamokos uždaviniai

- Susipažinsite su funkcija;
- Parašysite funkcijas, kurios grąžina apskaičiuotas reikšmes per funkcijos vardą;
- Pritaikysite sumos skaičiavimo algoritmą;
- Įtvirtinsite duomenų skaitymo iš failo ir rezultatų rašymo į failą įgūdžius.

Funkcijos

Fùnkcija yra vadinama programos konstrukcija, kuri atlieka savarankiškus veiksmus. Vykdant programą, į funkcijas galima kreiptis daug kartų. Funkcijos padeda struktūrizuoti programą. Tokią programą lengviau skaityti ir analizuoti.

Naujai kuriamų funkcijų aprašą galima skaidyti į dvi dalis: *prototipą* (išankstinį aprašą) ir realizavimo aprašymą.

Funkcijos prototipas nurodo naudotojui, kokia eilės tvarka duomenys perduodami funkcijai ir kaip gaunami rezultatai. Funkcijos prototipas rašomas prieš funkciją `main()` ir baigiamas kabliataškiu. Prototipo struktūra yra tokia:

```
rezultatoTipas funkcijosVardas(formaliejiParametrai);
```

Pavyzdžiui, `int Suma(int a, int b);`.

Funkcijos realizavimo aprašas (funkcijos antraštė ir veiksmai, kuriuos ji turi atlikti) pateikiamas už funkcijos `main()`. Jei funkcijos realizavimo aprašas yra prieš `main()`, tuomet nereikia rašyti funkcijos prototipo.

Funkcijos aprašas atrodo taip:

```
rezultatoTipas funkcijosVardas(formaliejiParametrai)
{
  funkcijosKamienas
}
```

Funkcijos antraštėje pirmiausia nurodomas grąžinamos reikšmės tipas. Jei funkcija per savo vardą jokios reikšmės negrąžina, vietoj `rezultatoTipas` nurodomas bazinis žodis `void`. Toliau rašomas `funkcijosVardas`, kuris parenkamas pagal tas pačias taisykles, kaip ir kintamųjų vardai. Po to skliaustuose išvardijami parametrai. Jei funkcija neturi parametrų, tuomet rašomi tušti skliaustai.

Funkcijos duomenis su programos duomenimis susieja `formaliejiParametrai`. Parametrai apibrėžiami pagal tas pačias taisykles, kaip ir kintamieji: nurodant jų reikšmės tipą ir vardą. Vienas nuo kito parametrai atskiriami kableliais.

Veiksmai, kuriuos turi atlikti funkcija, nurodomi dalyje `funkcijosKamienas`. Jei funkcija skirta kokiai nors reikšmei grąžinti, tai tarp funkcijos veiksmų turi būti bent vienas sakinys, kuriuo apskaičiuota reikšmė būtų grąžinama į programą. Funkcijos kamiene gali būti aprašomi kintamieji, reikalingi funkcijos veiksmams atlikti. Jie galioja tik funkcijoje.

Reikšmei grąžinti per funkcijos vardą naudojamas sakinys `return`. Sakinio `return` sintaksė yra tokia:

```
return Reiškinys;
```

`Reiškinys` – tai bet koks reiškinys, kurio reikšmė grąžinama atlikus veiksmą. Grąžinamos reikšmės tipas ir funkcijos antraštėje nurodytas rezultato tipas turi būti tarpusavyje suderinti. Sakiniu `return` ne tik grąžinama nurodyta reikšmė, bet ir nutraukiamas funkcijos darbas. Sakiniu `return;` tiesiog nutraukiamas funkcijos darbas.

Į funkcijas kreipiamasi jų vardais. Už jų skliaustuose pateikiami faktiniai parametrai (argumentai). Jei funkcija grąžina reikšmę, į ją kreipiamasi reiškiniuose, pavyzdžiui,

```
y = funkcijosVardas(faktiniaiParametrai) * c;
```

Jei funkcija grąžina apskaičiuotas reikšmes per parametrus (`void`), į ją kreipiamasi taip:

```
funkcijosVardas(faktiniaiParametrai);
```

Duomenis, su kuriais bus atliekami funkcijos veiksmai, nurodo `faktiniaiParametrai`. Tai gali būti reikšmės, kintamieji, reiškiniai.

Kreipinyje į funkciją faktinių parametrų paprastai būna tiek pat ir tokio pat tipo, kaip nurodyta funkcijos antraštėje. Parametrai rašomi tokia pat eilės tvarka, kaip nurodyta funkcijos antraštėje, ir tarpusavyje atskiriami kableliais. Jei funkcija parametrų neturi, kreipinyje rašomi tušti skliaustai.

Pavyzdys

```
// Stačiakampio plotas
#include <iostream>
using namespace std;
//-------------------------------------------------------------------------
int Plotas(int a, int b);          // funkcijos Plotas prototipas
//-------------------------------------------------------------------------
int main()
{
  int x = 5, y = 4, s;
  s = Plotas(x, y);                // kreipinys į funkciją Plotas
  cout << "Plotas = " << s << endl;
  return 0;
}
//-------------------------------------------------------------------------
// Skaičiuoja stačiakampio, kurio kraštinės a ir b, plotą
int Plotas(int a, int b) // funkcijos antraštė
{
  return a * b;            // per funkcijos vardą grąžinama apskaičiuota stačiakampio ploto reikšmė
}
```

Įvykdę programą, ekrane matysite:

```
Plotas = 20
```

Jei funkcija turi grąžinti keletą reikšmių, tuomet naudojami *parametrai-nuorodos*. Prieš juos funkcijos antraštėje rašomas ženklas `&`:

```
rezultatoTipas funkcijosVardas(tipas & vardas1, tipas & vardas2);
```

Tuo atveju, kai į funkciją kreipiamasi perduodant jai *parametrus-reikšmes*, funkcija sukuria naujus tų pačių tipų kintamuosius, kaip ir perduodami parametrai, ir jiems priskiria parametrų reikšmes. Vadinasi, funkcija dirba su parametrų reikšmių kopijomis, bet ne su pačiais parametrais. Tai patogu, kai funkcijai nereikia keisti parametrų reikšmių.

Tuo atveju, kai į funkciją kreipiamasi perduodant jai *parametrus-nuorodas*, ji gauna ne kintamųjų reikšmes, o nuorodas į kintamuosius (kintamųjų adresus). Vadinasi, funkcija tiesiogiai naudoja perduodamus kintamuosius.

Pateikiame pavyzdį, kuriame funkcija `Sukeisti()`, naudodama parametrų vardus `pirmas` ir `antras`, faktiškai naudojasi kintamaisiais `x` ir `y`.

Pavyzdys

```
//  Dviejų kintamųjų reikšmių sukeitimas
#include <iostream>
using namespace std;
//-------------------------------------------------------------------------
void Sukeisti(int & pirmas, int & antras);
//-------------------------------------------------------------------------
int main ()
{
  int x = 11, y = 25;
  Sukeisti(x, y);
  cout << " Kintamasis x po keitimo " << x << endl;
  cout << " Kintamasis y po keitimo " << y << endl;
  return 0;
}
//-------------------------------------------------------------------------
void Sukeisti(int & pirmas, int & antras)
{
  int papildomas = pirmas;
      pirmas = antras;
      antras = papildomas;
}
```

Įvykdę programą, ekrane matysite:

```
Kintamasis x po keitimo 25
Kintamasis y po keitimo 11
```

Užduotis

Mokyklos grindų dangos kaina. Mokykloje keičiama kabinetų grindų danga. Parenkite programą, kuri apskaičiuotų, kokia pinigų suma turi būti skirta naujai grindų dangai. Žinoma, kad kiekvienam kabinetui dangos reikia pirkti 3 proc. daugiau galimiems nuostoliams padengti. Apskaičiuotą pinigų sumą įrašykite į rezultatų failą.

Pirmoje pradinių duomenų failo eilutėje nurodytas kabinetų skaičius. Tolesnėse eilutėse įrašyta po tris realiuosius skaičius, atskirtus tarpais: kabineto ilgis, plotis ir vieno kvadratinio metro dangos kaina.

Pradinių duomenų ir rezultatų failų pavyzdys

Pradiniai duomenys	Paaiškinimai	Rezultatas	Paaiškinimai
3 4 5 54.60 3.5 4.5 35.55 9.2 3.3 42.63	Kabinetų skaičius Kabineto ilgis, plotis, dangos m^2 kaina Kabineto ilgis, plotis, dangos m^2 kaina Kabineto ilgis, plotis, dangos m^2 kaina	3034.54	Reikalinga pinigų suma

Algoritmas

Užduotis sprendžiama taip:

1. Nurodoma pinigų sumos, reikalingos visų kabinetų grindų dangai pakeisti, pradinė reikšmė (ji lygi nuliui).
2. Iš duomenų failo nuskaitomas kabinetų skaičius.
3. Veiksmai kartojami tiek kartų, kiek yra kabinetų:
   - iš duomenų failo perskaitomas kabineto ilgis, plotis ir vieno kvadratinio metro dangos kaina;
   - skaičiuojamas kabineto plotas;
   - skaičiuojama, kiek kainuos kabineto grindų danga;
   - prie visos pinigų sumos pridedama apskaičiuota kabineto grindų dangos kaina.
4. Į rezultatų failą įrašoma apskaičiuota pinigų suma, reikalinga visų kabinetų grindų dangai pakeisti.

Funkcijos kabineto plotui skaičiuoti rašymas

- Prieš pagrindinę funkciją `main()` parašykite funkcijos, skaičiuojančios stačiakampio plotą, prototipą:

```
double Plotas(double a, double b);
```

čia `a` yra stačiakampio ilgis, `b` – plotis.

- Po pagrindine funkcija `main()` parašykite funkciją `Plotas()`:

```
//---------------------------------------------------------------------
//  Apskaičiuoja stačiakampio, kurio ilgis a ir plotis b, plotą
double Plotas(double a, double b)
{
  double p;
  p = a * b;
  return p;
}
```

arba

```
//---------------------------------------------------------------------
//  Apskaičiuoja stačiakampio, kurio ilgis a ir plotis b, plotą
double Plotas(double a, double b)
{
  return a * b;
}
```

Šios funkcijos antraštėje yra du realiojo tipo kintamieji – a ir b. Jie atitinka stačiakampio ilgį ir plotį. Duomenų tipas double, parašytas prieš funkcijos vardą, parodo, kad apskaičiuotas plotas taip pat bus realusis skaičius. Funkcija gali būti užrašoma dviem būdais:

1) naudojant papildomą kintamąjį p ir per funkcijos vardą grąžinant apskaičiuotą kintamojo p reikšmę;
2) nenaudojant papildomo kintamojo, tik per funkcijos vardą grąžinant stačiakampio ilgio ir pločio sandaugą.

Pasirinkite tą būdą, kuris jums suprantamesnis.

- Patikrinkite, ar teisingai dirba sukurta funkcija: pagrindinėje funkcijoje parašykite, pavyzdžiui, tokį sakinį:

```
cout << Plotas(15, 10) << endl;
```

```
double Plotas(double a, double b);
//---------------------------------------------------------------------
int main()
{
  cout << Plotas(15, 10) << endl;
  return 0;
}
//---------------------------------------------------------------------
//  Apskaičiuoja stačiakampio, kurio ilgis a ir plotis b, plotą
double Plotas(double a, double b)
{
  return a * b;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti skaičių 150. Tai plotas stačiakampio, kurio ilgis lygus 15, o plotis – 10.

Parašytą funkciją panaudosime stačiakampio formos kabineto grindų plotui skaičiuoti, kai yra žinomas kabineto ilgis ir plotis.

Funkcijos, skaičiuojančios kabineto grindų dangos kainą, rašymas

- Prieš pagrindinę funkciją `main()` parašykite funkcijos, skaičiuojančios kabineto grindų dangos kainą, prototipą:

```
double DangosKaina(double p, double kaina);
```

čia `p` yra kabineto grindų plotas, `kaina` – vieno kvadratinio metro grindų dangos kaina.

- Po pagrindine funkcija `main()` parašykite funkciją `DangosKaina()`, skaičiuojančią pinigų sumą, reikalingą kabineto grindų dangai pirkti:

```
//-----------------------------------------------------------------
//  Apskaičiuoja kabineto grindų dangos kainą
//  p – kabineto plotas, kaina – dangos ploto vieneto kaina
double DangosKaina(double p, double kaina)
{
  double dk;
  dk = 1.03 * p * kaina;
  return dk;
}
```

arba

```
//-----------------------------------------------------------------
//  Apskaičiuoja kabineto grindų dangos kainą
//  p – kabineto plotas, kaina – dangos ploto vieneto kaina
double DangosKaina(double p, double kaina)
{
  return 1.03 * p * kaina;
}
```

Šios funkcijos antraštėje yra du realiojo tipo kintamieji – `p` ir `kaina`. Jie atitinka kabineto plotą ir vieno kvadratinio metro dangos kainą. Duomenų tipas `double`, parašytas prieš funkcijos vardą, parodo, kad apskaičiuota dangos kaina taip pat bus realusis skaičius. Funkcija gali būti užrašoma dviem būdais:

1) naudojant papildomą kintamąjį `dk` ir per funkcijos vardą grąžinant apskaičiuotą kintamojo `dk` reikšmę;
2) nenaudojant papildomo kintamojo, tik per funkcijos vardą grąžinant apskaičiuotą dangos kainą.

Pasirinkite tą būdą, kuris jums paprastesnis ir suprantamesnis.

- Patikrinkite, ar teisingai dirba funkcija: pagrindinėje funkcijoje parašykite, pavyzdžiui, tokį sakinį:

```
cout << DangosKaina(12.5, 10) << endl;
```

```
double Plotas(double a, double b);
double DangosKaina(double p, double kaina);
//---------------------------------------------------------------------
int main()
{
  cout << DangosKaina(12.5, 10) << endl;
  return 0;
}
//---------------------------------------------------------------------
//  Apskaičiuoja stačiakampio, kurio ilgis a ir plotis b, plotą
double Plotas(double a, double b)
{
  return a * b;
}
//---------------------------------------------------------------------
//  Apskaičiuoja kabineto grindų dangos kainą
//  p – kabineto plotas, kaina – dangos ploto vieneto kaina
double DangosKaina(double p, double kaina)
{
  return 1.03 * p * kaina;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti skaičių 128.75. Tai pinigų suma, reikalinga kabineto grindų dangai pirkti, kai kabineto grindų plotas yra 12,5 m^2, o dangos vieno m^2 kaina lygi 10.

Pradinių duomenų failo kūrimas, kintamųjų aprašymas

- Sukurkite tekstinį failą *Duomenys4.txt* ir į jį įrašykite pateiktus pavyzdyje pradinius duomenis.
- Aprašykite eilučių tipo konstantas pradinių duomenų ir rezultatų failų vardams atmintyje laikyti, funkcijų prototipus ir kintamuosius:

  `n` – kabinetų skaičius,
  `s` – visų kabinetų dangos kaina,
  `a` – kabineto ilgis,
  `b` – kabineto plotis,
  `k` – vieno kvadratinio metro dangos kaina.

```
const char CDfv[] = "Duomenys4.txt";      // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai4.txt";    // rezultatų failo vardas
//---------------------------------------------------------------------
double Plotas(double a, double b);
double DangosKaina(double p, double kaina);
//---------------------------------------------------------------------
int main()
{
  int n;                  // kabinetų skaičius
  double s = 0;           // pinigų suma, reikalinga visų kabinetų grindų dangai pirkti
  double a, b, k;         // kabineto ilgis, plotis ir grindų dangos vieno kv. metro kaina
  cout << DangosKaina(12.5, 10) << endl;
  return 0;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tokį pat rezultatą.

Pradinių duomenų skaitymas iš failo

- Pašalinkite iš programos sakinį, kuriuo buvo tikrinama, ar teisingai skaičiuojama kabineto dangos kaina:

```
cout << DangosKaina(12.5, 10) << endl;
```

- Papildykite programą sakiniu, kuriuo įvesties srautas `fd` susiejamas su failu *Duomenys4.txt*:

```
ifstream fd(CDfv);
```

- Parašykite sakinį kabinetų skaičiui perskaityti:

```
fd >> n;
```

- Parašykite sakinį, kuris patikrintų, ar reikšmė perskaityta teisingai:

```
cout << n << endl;
```

- Parašykite ciklo sakinį, kad būtų galima patikrinti, ar vieno kabineto pradiniai duomenys perskaityti teisingai:

```
for (int i = 1; i <= n; i++) {
  fd >> a >> b >> k;    // kabineto ilgis, plotis, dangos vieno kv. metro kaina
  cout << a << " " << b << " " << k << endl;
}
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti:

```
3
4 5 54.60
3.5 4.5 35.55
9.2 3.3 42.63
```

Kreipinių į funkcijas rašymas

- Pagrindinėje funkcijoje `main()` aprašykite realiojo tipo kintamąjį `p`:

```
double p;
```

- Pagrindinės funkcijos `main()` cikle esantį sakinį

  `cout << a << " " << b << " " << k << endl;`

  pakeiskite tokiais sakiniais:

```
p = Plotas(a, b);                                  // skaičiuojamas kabineto
                                                   // grindų plotas
cout << fixed << setprecision(2) << p << endl;     // rodomas ekrane apskaičiuotas
                                                   // kabineto grindų plotas
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti kiekvieno kabineto grindų plotą:

```
3
20.00
15.75
30.36
```

- Pagrindinę funkciją `main()` papildykite realiojo tipo kintamuoju `dk` (jį galite aprašyti ciklo viduje) ir parašykite sakinį:

```
dk = DangosKaina(p, k);   // pinigų suma, reikalinga kabineto grindų dangai pirkti
```

- Sakinį

  `cout << fixed << setprecision(2) << p << endl;`

  pakeiskite tokiu, kad ekrane būtų rodoma kiekvieno kabineto dangos kaina:

```
cout << fixed << setprecision(2) << dk << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti pinigų sumas, reikalingas kiekvieno kabineto grindų dangai pirkti:

```
3
1124.76
576.71
1333.07
```

6 Visų kabinetų grindų dangai reikalingos pinigų sumos skaičiavimas

- Norint apskaičiuoti pinigų sumą, kurios prireiks visų kabinetų grindų dangai įsigyti, reikia sudėti pinigų sumas, apskaičiuotas kiekvienam kabinetui. Papildykite pagrindinę funkciją `main()`: po kreipiniais į abi funkcijas parašykite sumos skaičiavimo sakinį:

```
s = s + dk;
```

Tuomet ciklo sakinys bus toks:

```
for (int i = 1; i <= n; i++) {
  double p, dk;
  fd >> a >> b >> k;
  p = Plotas(a, b);
  dk = DangosKaina(p, k);
  s = s + dk;
}
```

- Sumos skaičiavimą galima užrašyti vienu sakiniu:

```
s = s + DangosKaina(Plotas(a, b), k);
```

arba

```
s += DangosKaina(Plotas(a, b), k);
```

Tuomet ciklo sakinys taps trumpesnis:

```
for (int i = 1; i <= n; i++) {
  fd >> a >> b >> k;
  s = s + DangosKaina(Plotas(a, b), k);
}
```

arba

```
for (int i = 1; i <= n; i++) {
  fd >> a >> b >> k;
  s += DangosKaina(Plotas(a, b), k);
}
```

- Tarpinių skaičiavimo rezultatų rodymo ekrane sakinius pašalinkite, o už ciklo parašykite tokį sakinį:

```
cout << fixed << setprecision(2) << s << endl;
```

- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti apskaičiuotą pinigų sumą, reikalingą visų kabinetų grindų dangai įsigyti:

```
3034.54
```

Rezultatų rašymas į failą *Rezultatai4.txt*

- Papildykite programą sakiniu, kuriuo išvesties srautas `fr` susiejamas su failu *Rezultatai4.txt*:

```
ofstream fr(CRfv);
```

- Parašykite apskaičiuotos pinigų sumos rašymo į failą sakinį:

```
fr << fixed << setprecision(2) << s << endl;
```

- Kai visi rezultatai bus surašyti, srautą užverkite:

```
fr.close();
```

- Pašalinkite sakinius, kurie skaičiavimo rezultatus rodydavo ekrane.
- Įrašykite ir įvykdykite programą. Ekrane turėtumėte matyti tik informacinį pranešimą.
- Atverkite rezultatų failą *Rezultatai4.txt*. Jame turėtų būti skaičius 3034.54 – pinigų suma, reikalinga grindų dangai įsigyti.

Pagrindinė funkcija

Nuosekliai atlikus visus šio darbo žingsnius, pagrindinė funkcija turėtų būti tokia:

```
const char CDfv[] = "Duomenys4.txt";     // pradinių duomenų failo vardas
const char CRfv[] = "Rezultatai4.txt";   // rezultatų failo vardas
//---------------------------------------------------------------------------
double Plotas(double a, double b);
double DangosKaina(double p, double kaina);
//---------------------------------------------------------------------------
int main()
{
  int n;                 // kabinetų skaičius
  double s = 0;          // pinigų suma, reikalinga visų kabinetų grindų dangai pirkti
  double a, b, k;        // kabineto ilgis, plotis ir grindų dangos vieno kv. metro kaina
  ifstream fd(CDfv);
  fd >> n;
  for (int i = 1; i <= n; i++) {
    fd >> a >> b >> k;
    s += DangosKaina(Plotas(a, b), k);
  }
  fd.close();
  ofstream fr(CRfv);
  fr << fixed << setprecision(2) << s << endl;
  fr.close();
  return 0;
}
```

Programos patikrinimas

- Patikrinkite, kaip dirba programa, kai:
  - mokykla turi tik vieną kabinetą;
  - pradinių duomenų faile kabineto plotis ir ilgis surašomi bet kuria eilės tvarka;
  - pirmoje failo eilutėje kabinetų skaičius yra 0 (nulis).

Užduotys

Dangoraižis

Naujame dangoraižyje yra n kabinetų, sunumeruotų nuo 1 iki *n*. Prie kiekvieno kabineto durų reikia prikalti lenteles su to kabineto numeriu. Ant vienos lentelės užrašytas tik vienas skaitmuo. Jei kabineto numeris nevienženklis skaičius, tai prie durų kalamos kelios lentelės. Parenkite programą, kuri apskaičiuotų, kiek reikės lentelių, norint prikalti numerius prie visų dangoraižio kabinetų durų.

Pradinių duomenų faile įrašyta, kiek kabinetų yra dangoraižyje. Rezultatų faile turi būti įrašytas lentelių skaičius.

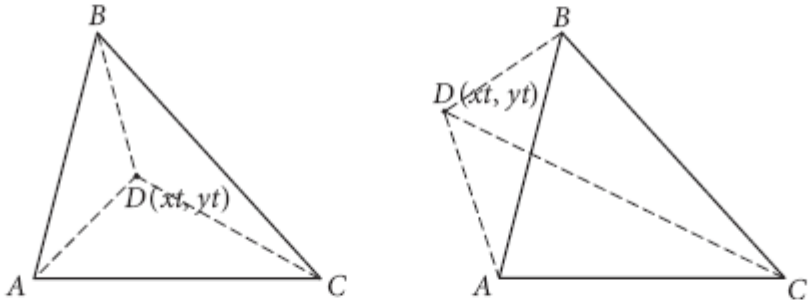
Pradinis duomuo	Rezultatas
16	23

Užduotys

Trikampis ir taškas

Stačiakampėje koordinačių plokštumoje nubraižytas trikampis, kurio viršūnių koordinatės yra žinomos. Plokštumoje padėtas taškas, kurio koordinatės yra (*xt*, *yt*). Parenkite programą, kuri nustatytų, ar taškas yra trikampio viduje, ar jo išorėje. Atstumas tarp dviejų taškų (*x1*, *y1*) ir (*x2*, *y2*) yra skaičiuojamas pagal formulę $atst = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$.

Jeigu taškas yra trikampio viduje, tai, tašką sujungus su trikampio viršūnėmis, gautų trikampių plotų suma bus lygi duoto trikampio plotui. Jeigu išorėje, tuomet gautų trikampių plotų suma bus didesnė už duoto trikampio plotą.



Kai taškas yra trikampio viduje, tai PlotasABC = PlotasABD + PlotasBCD + PlotasACD.

Visi skaičiavimai atliekami tam tikru tikslumu, nes duomenys yra realieji skaičiai (`double` tipo).

Pasitikrinkite. Kai *A*(*10, 20*), *B*(*20, –5*), *C*(*–5, –4*), *xt* = *5*, *yt* = *5*, ekrane turi būti rodomas pranešimas:

```
Taškas yra trikampio viduje.
```

Kai *xt* = *–6*, *yt* = *9*, ekrane turi būti rodomas pranešimas:

```
Taškas yra trikampio išorėje.
```

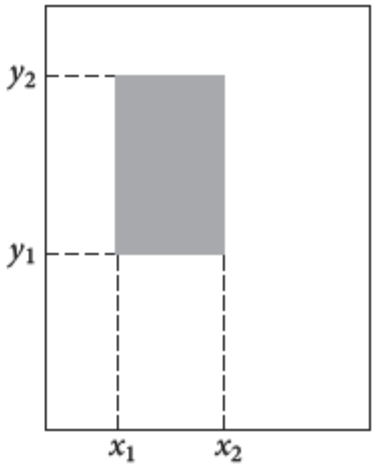
Sunkesnės užduotys

Siena

Tomas kolekcionuoja plakatus. Vieną dieną jis sugalvojo visus turimus plakatus suklijuoti ant savo kambario sienos. Šiek tiek pamąstęs, jis popieriaus lape sužymėjo tikslias plakatų vietas ant sienos. Tomas nori matyti visą savo kolekciją, todėl stengėsi, kad nė vienas plakatas neuždengtų kitų. Visi plakatai yra stačiakampiai ir pagal Tomo sudarytą planą jų kraštinės turi būti lygiagrečios su sienos šonais.

Tomo mama susirūpinusi – neseniai dažyta siena dabar bus uždengta plakatais. Jai įdomu, kiek sienos liks neuždengta. Parenkite programą, kuri apskaičiuotų nepaslėptos po plakatais sienos dalies plotą. Primename, kad stačiakampio plotas lygus jo kraštinių ilgių sandaugai.

Pradiniai duomenys pateikti faile. Pirmoje failo eilutėje įrašyti du skaičiai – sienos plotis p ir aukštis a centimetrais ($1 \le p, a \le 1000$). Antroje eilutėje įrašytas vienas sveikasis skaičius N ($1 \le N \le 1000$) – plakatų kiekis. Tolesnėse N eilučių įrašyta po keturis sveikuosius skaičius x_1, y_1, x_2, y_2 ($0 \le x_1 < x_2 \le p$; $0 \le y_1 < y_2 \le a$): x_1 – atstumas nuo kairiojo sienos krašto iki kairiosios plakato kraštinės; y_1 – atstumas nuo apatinio sienos krašto iki apatinės plakato kraštinės; atitinkamai x_2 ir y_2 – atstumai iki plakato dešiniosios ir viršutinės kraštinių. Visi atstumai nurodyti centimetrais.



Į rezultatų failą įrašykite vieną skaičių – neuždengtos plakatais sienos dalies plotą kvadratiniais centimetrais.

Pradiniai duomenys	Rezultatas
300 200 2 50 100 150 150 200 0 300 100	45000

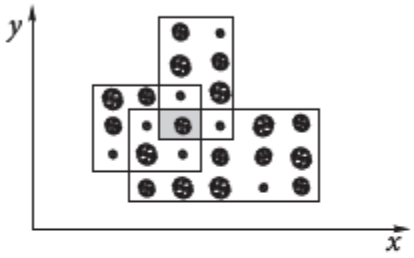
(XVIII olimpiada, 2006)

Sunkesnės užduotys

Trys sodininkai

Trys draugai, apsigyvenę kaime, nusprendė mokytis sodininkauti. Kaime buvo didžiulis sodas, kurio kiekviename vienetiniame plotelyje augo po vieną vaismedį.

Kiekvienas iš trijų draugų pasirinko stačiakampį sklypą ir nusprendė prižiūrėti jame esančius medžius. Susirinkus draugėn paaiškėjo, kad jų pasirinkti sklypai įsiterpia vienas į kitą, t. y. kai kuriuos vaismedžius prižiūrės ne vienas, o keletas sodininkų. Parenkite programą, kuri apskaičiuotų, kiek vaismedžių panorėjo prižiūrėti visi trys draugai.



Pradiniai duomenys pateikiami trijose duomenų failo eilutėse. Į kiekvieną eilutę įrašyta po keturis skaičius, apibūdinančius kiekvieno draugo pasirinktą sklypą: sklypo apatinio kairiojo ir viršutinio dešiniojo kampų koordinatės (pirma koordinatė x, po to – y). Visos koordinatės – sveikieji skaičiai. Rezultatų faile turi būti įrašytas vienas skaičius – kiek vaismedžių panorėjo prižiūrėti visi trys draugai.

Pradiniai duomenys	Rezultatas
30 30 80 70 10 20 70 90 50 20 100 90	800

(X olimpiada, 1999)