МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2020
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2020
Proceedings of the Forty-ninth Spring Conference
of the Union of Bulgarian Mathematicians
2020

ГЕОДЕЗИЧНИ ЗАДАЧИ В ОБУЧЕНИЕТО ПО ПРОГРАМИРАНЕ

Владимир Заблоцкий, Светлана Василева

Представена е учебна програма за студенти картографи и геодезисти, изучаващи основи на програмирането на език C++, а също и ученици, обучаващи се в профил математика и интересуващи се от науките за Земята. В примера се изчисляват плоски правоъгълни координати на търсена точка чрез обратно геодезично засичане с допълнителна контролна точка (задача на Потено[1]). В програмния фрагмент задачата се решава чрез свеждане до задачата за пряко засичане с използване на формулите на Деламбр[2]. Програмата изчислява и извежда на стойностите на декартовите координати на търсената точка, ако разминаването между абсцисите и ординатите на двете решения удовлетворява правилото за „трите сигми“. В примера са използвани технологиите за процедурно програмиране, което прави подобни задачи подходящи за учебни проекти в часовете по Програмиране (C++) за профил *математика* (*информатика*).

Увод. Информатиката стана много популярна дисциплина, преподавана както в средното (и началното училище), така и във висшите училища. Това, което различава информатиката от другите научни дисциплини, е, че се появи сравнително неотдавна и няма дълбоки исторически корени, характерни за почти всички академични науки. В днешно време става очевидно, че съществените изменения в методите за обучение са свързани с внедряването на новите информационни технологии в образователния процес. Компютърните мрежи направиха дистанционното образование много по-достъпна форма за придобиване на знания. Използването на мултимедийни проектори и интерактивни дъски в университетските и училищните аудитории, на персонални и мобилни компютри в лабораторните зали също способства за промените в начините на преподаване на информатиката. Ако преди преподавателят е трябвало да пише на (тебеширената) дъска само отделни програмни фрагменти, то днес с помощта на мултимедиен проектор, на обучаемите се демонстрира целия програмен код. Освен това използването на демонстрационен софтуер позволява да се покаже на аудиторията работата на програма в реално време.

Обикновено студентите и учениците изучават програмиране по учебници, предназначени за обширна аудитория („за подпомагане на обучението по информатика“). Такива учебници, например за езика C++, има много: класически учебници – преводни (на С. Прата [1] и много други), български и руски (на В. Подбельский [2], М. Тодорова [3] и много други); самоучители (като този на Дж. Либерти

[1] известна още като задача на Снелиус-Потено (Snellius-Pothenot Problem) – бел. ред.
[2] Jean-Baptiste-Joseph Delambre (1749—1822) – френски математик и астроном – бел. ред.

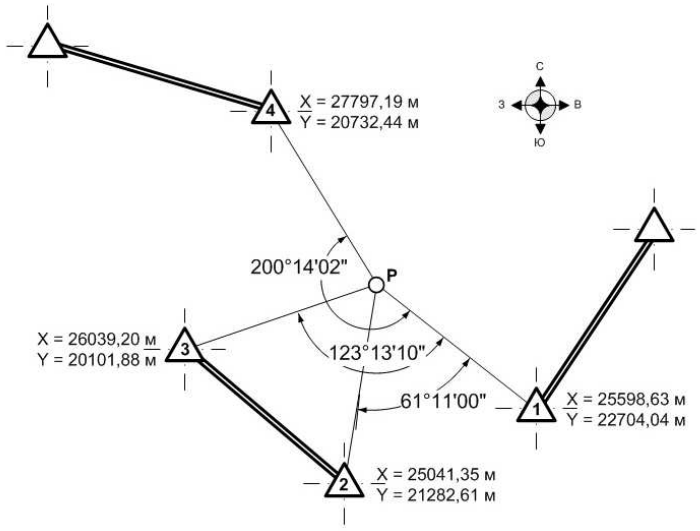
[4] и други); учебници по информатика (за 9. клас, Програмиране на C++ [5] и други). Но всички тези издания са за широк кръг читатели и затова в тях няма програми и примери, специализирани за профилирани паралелки в средното училище или профилни инженери, например картографи и геодезисти. Целенасоченото и специализирано обучение по програмиране трябва да отчита направлението на подготовката на учениците и студентите, за да им даде знания, които ефективно ще прилагат в практиката си. Авторите предлагат обучението по програмиране да бъде свързано с тематичната насоченост на бъдещите специалисти: например, да се обучават студентите картографи и геодезисти на основата на едни компютърни програми и задачи, а например студентите и учениците (профил *физика*) – с други задачи и компютърни програми, машиностроителните специалности – по трети тип и т.н.

Целта на работата е да предложи тип геометрични – геодезични задачи [6], които могат да бъдат част от обучението по информатика за студенти и ученици (11.–12. клас, обучаващи се в профил *математика*) – алгоритмично програмиране и евентуално визуално програмиране. Такива задачи, но по информационни технологии, са реализирани в [7]. Показаният примерен тип задачи подпомага студенти (бъдещи инженери геодезисти и картографи) в решаването на специфичните професионални проблеми ([8], [9]) много по-ефективно, отколкото общите типови задачи, с които се изучава програмиране.

Решаване на задачата за обратното геодезично засичане с помощта на контролна точка. Представената компютърна програма за студенти картографи и геодезисти, демонстрира решението на обратното ъглово засичане по метода на Деламбр на езика за програмиране C++ (изучаван в много университети в България, Русия и много други европейски държави).

Геодезична постановка на задачата: Необходимо е да се определят координатите на точка $P(x_P, y_P)$, ако са известни декартовите координати на три начални точки и координатите на четвърта контролна точка. Предполага се също, че се използва ъглов измервателен прибор (теодолит), намиращ се в точката, чиито координати трябва да се определят. Тази задача е била формулирана и решена за първи път от френския математик Потено, затова определянето на координатите на точка по метода на обратното засичане се нарича *задача на Потено*. В днешно време има публикувани много алгоритми за решаване на задачата на Потено, един от които е методът на Деламбр. Същността на алгоритъма се състои в привеждането на решението на обратното засичане към решението на прякото засичане по формулите на Гаус. За тази цел е необходимо по началните входни данни да се определят дирекционните ъгли на лъчите от началните точки към определяната точка. В работата се разглежда съставената авторска C++ програма за решаване на задачата на Потено по метода на Деламбр.

Формулировка на задачата, определяне на началните стойности и означения. Нека координатите на началните точки са: $1(x_1, y_1)$, $2(x_2, y_2)$, $3(x_3, y_3)$ и $4(x_4, y_4)$ (фиг. 1). Трябва да се определят координатите на точка $P(x_P, y_P)$. Измерени и известни са също и три хоризонтални ъгъла: ъгъл β_1 между лъча към точка 1 и лъча към точка 2, ъгъл β_2 между лъча към точка 1 и лъча към точка 3 и ъгъл β_3 между лъча към точка 1 и лъча към точка 4. На фиг. 1 е показана схемата на обратното ъглово засичане с допълнителна контролна точка.



Фиг. 1. Схема на обратното ъглово засичане: опорни точки 1, 2, 3, допълнителна контролна точка 4, търсена точка

Изчисляването на координатите x_P и y_P на определяемата точка P се извършва по схемата, описана с формули (1)–(9). Отначало се изчислява α_1 – дирекционният ъгъл на началната посока от точка P към точка 1 по формула (1) [8] и [9].

$$\operatorname{tg}\alpha_1 = \frac{(y_2 - y_1)\operatorname{cotg}\beta_1 + (y_1 - y_3)\operatorname{cotg}\beta_2 - x_2 + x_3}{(x_2 - x_1)\operatorname{cotg}\beta_1 + (x_1 - x_3)\operatorname{cotg}\beta_2 - y_2 + y_3} \tag{1}$$

След това се изчисляват дирекционните ъгли на направленията от началните точки 1, 2 и 3 към точка P по формули (2)–(5) [8] и [9]:

$$\alpha_{1-P} = \alpha_1 \pm \pi, \tag{2}$$

$$\alpha_{2-P} = \alpha_{1-P} + \beta_1, \tag{3}$$

$$\alpha_{3-P} = \alpha_{1-P} + \beta_2, \tag{4}$$

$$\alpha_{4-P} = \alpha_{1-P} + \beta_3, \tag{5}$$

където α_{1-P}, α_{2-P}, α_{3-P}, и α_{4-P}, са съответно дирекционните ъгли от началните точки 1, 2, 3 и 4 към точка P.

След това координатите на търсената точка P се определят в два екземпляра по формули (6), (7) и (8), (9) [8] и [9]:

$$x'_P = \frac{x_1 \cdot \operatorname{tg}\alpha_{1-P} - x_2 \cdot \operatorname{tg}\alpha_{2-P} + y_2 - y_1}{\operatorname{tg}\alpha_{1-P} - \operatorname{tg}\alpha_{2-P}}, \tag{6}$$

$$y'_P = y_1 + (x'_P - x_1) \cdot \operatorname{tg}\alpha_{1-P}, \tag{7}$$

където x_1, x_2 са абсцисите на точки 1 и 2, y_1, y_2 – ординатите на точки 1 и 2.

$$x''_P = \frac{x_3 \cdot \operatorname{tg}\alpha_{3-P} - x_4 \cdot \operatorname{tg}\alpha_{4-P} + y_4 - y_3}{\operatorname{tg}\alpha_{3-P} - \operatorname{tg}\alpha_{4-P}}, \tag{8}$$

(9) $$y''_P = y_4 + (x''_P - x_4) \cdot \operatorname{tg} \alpha_{4-P},$$

където x_3, x_4 са абсцисите на точки 3 и 4, y_3, y_4 – ординатите на точки 3 и 4, $\operatorname{tg} \alpha_{3-P}$.

При изчисляването на координатите на точка P се прилага правилото, при което изходните точки се номерират по часовниковата стрелка, отчитайки, че наблюдателят се намира в точка P. В изчисленията се получават две двойки декартови координати. Разликите между абсцисите и ординатите на тези стойности трябва да удовлетворяват неравенството, наречено „правило на трите сигми“ [9] (формули (10)–(11)).

(10) $$r = \sqrt{(x''_P - x'_P,)^2 + (y''_P - y'_P)^2} < 3M_r,$$

където

(11) $$M_r = \sqrt{M_1^2 + M_2^2}.$$

Стойностите на M_1 и M_2, средноквадратичните грешки за положението на точка P, определено по четирите изходни точки (1, 2, 3 и 2, 3, 4), се изчисляват по формули (12) и (13):

(12) $$M_1 = \frac{d' \cdot m_\beta}{\rho \sin(\varphi'_1 + \varphi''_2)} \sqrt{\frac{d_1^2}{a^2} + \frac{d_2^2}{b^2}},$$

(13) $$M_2 = \frac{d'' \cdot m_\beta}{\rho \sin(\varphi'_1 + \varphi''_2)} \sqrt{\frac{d_1^2}{a^2} + \frac{d_2^2}{b^2}},$$

където: m_β е средноквадратичната грешка при измерване на ъгъла; φ'_1, φ'_2 и φ''_1, φ''_2 – ъглите при точки 1 и 3 и съответно при точки 2 и 4; d', d'' – разстоянията от точка P до точки 2 и 3; a и b – разстоянията от точка 1 до точка 2 и съответно от точка 2 до точка 3.

Когато има допустими разминавания между координатите, крайните стойности на координатите на точка P се изчисляват като средноаритметично от двете стойности.

Програма за определяне на неизвестните координати по метода на Деламбр. На фиг. 2 е даден основен фрагмент от учебната програма за определяне на неизвестните координати на точка с обяснения за значението на сегментите. Използвани са стандартните математически функции за тангенс `tan` и за аркустангенс `atanl` (с увеличена точност) и е написана функция за изчисляване на котангенс `1/tan`.

Грешката в положението на точка P се определя чрез разстоянията, изчислени в масив `distance[i]`.

(14) $$r = \sqrt{(x_1 - x_2)^2 + (y_1 -, y_2)^2},$$

(15) $$r = \sqrt{x_1^2 + y_1^2}.$$

06–07: *Функция PythagorasFormula* – пресмятане на разстояния по Питагоровата формула.
Ако във функцията се предават четири параметъра, т.е x_1, y_1 – координати на първата точка и x_2, y_2 – координати на втората точка, то изчисляваме евклидовото разстояние м/у точките по формула (14); ако във функцията се предават само два параметъра, например x_1, y_1, то другите два параметъра по подразбиране са $x_2 = 0$ и $y_2 = 0$ и се изчислява евклидово разстояние м/у точката и началото на координатната система по формула (15).

```
06: double PythagorasFormula(double x1,double y1,double x2 = 0,double y2 = 0)
07: {return sqrt(pow((x1-x2),2) + pow((y1- y2),2));}
```

11–14: Дефинираме: *масиви* `x[4], y[4]` за съхраняване на координатите на четирите начални точки,
`xP[2], yP[2]` – за координатите на определяемата точка, изчислявани при едната комбинация на началните точки и при другата;
`distance[4]` – за разстоянията от определяната т. P до четирите начални точки.
12: *променливи* `XP, YP` – за средната стойност на координатите на определяемата точка;
`degrees`, `minutes`, `seconds` – за въвеждане на хоризонталните ъгли отделно в градуси, минути и секунди;
`mb` – за средноквадратичната грешка при измерване на ъгъл.
14: *масиви* `angle[3]` – за измерените при т. P ъгли във формат градуси с дробна част, `angleInRadian[3]` – за същите ъгли, но в радиани;
`ctgB[i]` – за котангенсите на ъглите;
`azimuthN_P[4]` – за дирекционните ъгли м/у линиите `1P, 2P, 3P` и `4P`, и линиите, съединяващи точки 1, 2, 3 и 4 с т. P.

```
10: int main()
11: { double x[4],y[4],xP[2],yP[2];
12:    double XP,YP,distance[4],degrees, minutes,seconds, mb;
14:    double angle[3], angleInRadian[3],ctgB[3],azimuthN_P[4];
16:    // Въвеждане координатите X, Y на точки 1-3
19:
```

21–26: *Въвеждане* големините на трите ъгъла в градуси, минути и секунди, *преобразуване* на ъглите във формат градуси с цяла и дробна част, и след това в радиани.
26: изчисляване котангенсите на ъглите и съхраняване резултатите в масив `ctgB[3]`.

```
21: for(int i = 0; i < 3; i++) {
22:  cout «"Въведете ъгъла между линия(P-1) и линия(P-≪< i+2«"), град.мин.сек:
";
23: cin » degrees » minutes » seconds;
24: angle[i] = degrees + minutes/60 + seconds/3600;
25: angleInRadian[i] = angle[i] * M_PI/180;
26: ctgB[i] = 1/tan(angleInRadian[i]);}
```

28: Изчисляване тангенс на дирекционния ъгъл на началното направление (лъчът от т. P към т. 1 съгласно формула (1).
29: Пресмятане големината на дирекционния ъгъл на началната посока чрез функция аркустангенс `atanl` (в радиани). Ако се получава отрицателна стойност, към големината на ъгъла се добавя 2π.

```
28: double tgA1_P = ((y[1]-y[0])*ctgB[0]+(y[0]-y[2])*ctgB[1]-x[1]+x[2])/((x[1]-
     x[0])*ctgB[0]+(x[0]-x[2])*ctgB[1]+y[1]-y[2]);
29: azimuthN_P[0] = atanl(tgA1_P);
```

31–34: Пресмятане дирекционните ъгли на вектори 1, 2, 3, 4 по формули (2) и (3).

33: Ако големината на дирекционния ъгъл е по-голяма от 360°, то се изважда 2π.

```
30: if(azimuthN_P[0] < 0) azimuthN_P[0] +=2*M_PI;
31: for(int i = 0; i < 3; i++){
32: azimuthN_P[i+1]= azimuthN_P[0]+angleInRadian[i];
33: if(azimuthN_P[i+1] > 2*M_PI) azimuthN_P[i+1] -=2*M_PI;
34: }
```

35–37:Изчисляване на X, Y-координатите на допълнителната точка P по формули (4) и (5);

38: И извеждане на резултата.

```
35: for(int j = 0,i = 0; i < 3; j++, i = i+2){
36: xP[j]=(x[i]*tan(azimuthN_P[i])-x[i+1]*tan(azimuthN_P[i+1])+y[i+1]-
    y[i])/(tan(azimuthN_P[i])-tan(azimuthN_P[i+1]));
37: yP[j] = y[i]+(xP[j]-x[i])*tan(azimuthN_P[i]);
38: cout «"X-координата на т. P«<j«": «< xP[j] «"m«< endl«"Y-координата на
    т. P«<j«": «< yP[j] «"m«< endl;
39:   }
```

40–83: Оценка на точността на даденото обратно геодезично засичане.

Променлива `mb` – средноквадратична грешка при измерване на ъгъла (в примера – $2''$)

променлива `ro = 206265` – брой секунди в 1 радиан (служи за преобразуване на грешката от радиани в ъглови секунди.

46: *Извикване* на ф-цията за изчисляване на разстояние (редове *06–07*) с четири параметъра – координатите на двете точки, едната е т. P.

```
40:cout«"Средноквадратична грешка за измерване на ъгъла: ";
41: cin » mb;
42: int ro = 206265;
43: double lengthBaseLine [3], azimuthBaseLine[3];
44: double deltaY[3], deltaX[3], fi[4], M_P[2];
45: for(int i = 0; i < 4; i++)
46: distance[i] = PythagorasFormula(x[i],y[i],xP[0],yP[0]);
```

48–49: Изчисляване в цикъл дължините на трите базисни линии (*масив* `lengthBaseLine[3]`) с Питагорова формула – към функцията `PythagorasFormula` се предават четири параметъра (координатите на две съседни точки).

```
48: for(int i = 0; i < 3; i++)
49: lengthBaseLine[i] = PythagorasFormula(x[i],y[i],x[i+1],y[i+1]);
```

51–55: Изчисляват се азимутите на базисните линии: изчисляване нарастването на координатите на линиите `deltaY[i]` и `deltaX[i]`;

54: определяне азимута на базисната линия с вградената C++ функция аркустангенс `atanl`. Ако се получи отрицателна стойност, се добавя 2π.

```
51: for(int i = 0; i < 3; i++){
52:    deltaY[i] = y[i+1]-y[i];
53:    deltaX[i] = x[i+1]-x[i];
54:    azimuthBaseLine[i] = atan2l(deltaY[i],deltaX[i]);
55:    if (azimuthBaseLine[i] < 0) azimuthBaseLine[i] += 2*M_PI;}
```

57–69: При оценката на грешките участват големините на ъгли φ_1 и φ_2. Ъглите се намират като разлика от азимутите на посоките на линиите. За φ_1 – формулата е φ_1=`azimuthN_P[i]- azimuthBaseLine[i]` (от азимута на линията, съединяваща т. P и началната т. 1 се изважда азимутът на базисната линия 1–2), а φ_2=`azimuthBaseLine[i-1]-` π `- azimuthN_P[i]` (от азимута на базисната линия 2–3 се

изважда π и се получава обратният азимут на линия 2–3 (линия 3–2) и се изважда азимутът на линията, съединяваща т. P и нач. т. 3.

58–68: Избор на формула за разчет на ъглите ϕ_1 и ϕ_2. Винаги получените величини се проверяват за допустимост на значенията на дирекционните ъгли, които не трябва да са отрицателни и не трябва да са по-големи от 360°. Ако тези условия не се изпълняват, получените стойности се привеждат към допустимия диапазон за ъгли $(0^\circ - 360^\circ)$.

```
57: for(int i = 0; i < 4; i++){
58:    if (i < 2) {
59:      fi[i]=azimuthN_P[i]- azimuthBaseLine[i];
60:      if (fi[i] < 0) fi[i] += 2*M_PI;
61:      if (fi[i] > 2*M_PI) fi[i] -= 2*M_PI;}
63:    else
64:    { fi[i] = azimuthBaseLine[i-1] - M_PI - azimuthN_P[i];
66:    if (fi[i] < 0) fi[i] += 2*M_PI;
67:    if (fi[i] > 2*M_PI) fi[i] -= 2*M_PI;
68:    }
69: }
```

70–71: Изчисляване на грешката при определяне положението на т. P съгласно формули (12), (13) по трите начални точки за първата комбинация точки и за втората комбинация. В примера се използва първата комбинация начални точки 1, 2, 3 и втората комбинация точки 2, 3, 4.

```
70: for(int i = 0; i < 2; i++)
71: M_P[i] = (distance[i+1]*mb)/(ro*sin(fi[i] + fi[i+2]))*sqrt(pow(distance[i]
   / lengthBaseLine[i],2) +pow(distance[i+2]/lengthBaseLine[i+1],2));
```

73: Изчисляване общата средноквадратична погрешност за положението на точка по формула (15) чрез функцията `PythagorasFormula`. Разминаванията в стойностите на координатите на определяната точка P, не трябва да надхвърлят „трите сигми“.

```
73: double M = PythagorasFormula(M_P[0],M_P[1]);
74: double r = PythagorasFormula (xP[0],yP[0],xP[1],yP[1]);
```

76–77: Ако намерените два варианта координати удовлетворяват правилото на „трите сигми“, то разликата се приема за допустима и X и Y-координатите на т. P се изчисляват като средноаритметични на двете стойности.

78: Извеждане на екран на изчислените координати на т. P.

```
75: if (r <= 3*M){
76:    XP = (xP[0] + xP[1])/2;
77:    YP = (yP[0] + yP[1])/2;
78:   cout «"X-coordinate of point P: «<XP«"m\n«<"Y-coordinate of point P:
     «<YP«"m\n";
79: }
```

81: Иначе се извежда съобщение, че точността на ъгловите измервания не е достатъчна.

```
80: else
81:    { cout«"Точността на ъгловите измервания не е достатъчна: «< endl;}
84: return 0;
85: }
```

Фиг. 2. C++ програмен модул за определяне на неизвестни координати по метода на Деламбр

Резултати. Нека са въведени примерните данни (по [9]), представени в таблица 1.

Таблица 1. Тестови начални данни за програмата за обратно ъглово засичане с контролна точка

Пункт	Хоризонтални ъгли при т. P	Абциса	Ордината
1	$0^\circ 0' 00''$	25598,63	22704,04
2	$61^\circ 11' 00''$	25041,35	21282,61
3	$123^\circ 13' 10''$	26039,20	20101,88
4	$200^\circ 14' 02''$	27797,19	20732,44

Като резултат от работата на програмата (фиг. 2) на екрана се извежда следното:
X-координата на точка P: 26522.12 м
Y-координата на точка P: 21521.99 м

Заключение и бъдеща работа. Демонстрирана е предметно-ориентирана учебна програма на език C++ с геодезично съдържание. Примерната програма може да се използва както при студенти геодезисти и картографи, така и за ученици (11–12 клас), обучаващи се в природоматематически гимназии и професионални гимназии по архитектура, строителство и геодезия. Програмата изчислява плоски правоъгълни координати на допълнителна точка чрез решаване на обратното ъглово геодезично засичане (т. нар. *Задача на Потено*). Входните данни за програмата са координатите на четирите начални точки и големините на трите ъгъла, измерени в търсената точка в посоките от т. 1 до т. 2, от т. 1 до т. 3 и от т. 1 до т. 4. Програмата изчислява координатите на определяната точка по формулите на Деламбр и извежда на екран получените координати чрез използване на технологиите на процедурното програмиране.

Използването на подобни програми в часовете по информатика или като курсови задачи (за самостоятелна работа) са изключително полезни и ефективни, както като учебно средство на преподавателя по информатика и специализираните (профилиращи) предмети в математическите и техническите гимназии и университетите, така и като начин за самостоятелно овладяване на практически знания и умения от ученика и студента, изучаващ програмиране. Като едно продължение на представената работа, авторите си поставят за задача „да внедрят“ примерната програма в часовете по програмиране в 11. и 12. клас на профил *математика* в природоматематическа гимназия и да обобщят и анализират получените резултати.

Такъв тип задачи могат да бъдат реализирани и с други съвременни алгоритмични езици, както и в среди като Visual Studio и други, което предоставя огромно множество възможности за формулиране от преподавателите на най-разнообразни задания за самостоятелна работа на ученици (профил *математика* и профил *софтуерни и хадуерни технологии*) и студенти не само по картография и геодезия, но и по други университетски специалности, в които науките за Земята са от първостепенно значение за бъдещата професия.

ЛИТЕРАТУРА

[1] С. Прата. Язык программирования C++. Лекции и упражнения. Москва, ООО „ИД Вильямс“, 2014.

[2] В. Подбельский. Язык Си++. Москва, Финансы и статистика, 2003, 560 стр.

[3] М. Тодорова. Програмиране на C++, ч. 1, изд. Сиела, 2004, 378 с.
[4] Дж. Либерти. (2004) Освой самостоятельно C++ за 21 день. 1 Москва, ООО „ИД Вильямс“, https://vakho99.files.wordpress.com/2013/01/liberti_c__za_21_den.pdf (10.12.2019)
[5] Л. Иванова, В. Вазова, Б. Танева, И. Стоянов. Информатика за 9 клас: Програмиране на C++, 2007, Пловдив, Коала Прес.
[6] В. Заблоцкий, С. Василева. Многомодульная учебная программа, моделирующая измерение теодолитом расстояний, горизонтальных и вертикальных углов, *Известия высших учебных заведений „Геодезия и аэрофотосъемка“*, **62**, *6* 2018, 632–642. https://geocartography.ru/source/izvestia_vuzov/2018_6_632-642
[7] С. Василева, Е. Николова. Електронни таблици в обучението по агрономическите дисциплини. *Списание за наука Ново знание*, II, 1, Изд. на Висше училище по агробизнес и развитие на регионите, Пловдив, 2013, 298–307, https://uard.bg/files/custom_files/files/documents/New%20knowledge/year2_n1/paper_svvasileva_y2n1.pdf
[8] А. Маслов, А. Гордеев, Ю. Батраков. Геодезия, Москва, Изд. Колоср 2006.
[9] Г. Поклад, С. Гриднев. Геодезия: учебное пособие для вузов. Москва, Академический Проектр, 2007.

Владимир Заблоцкий
Московски държавен университет по геодезия и картография
Гороховский пер., 4
105064, Москва, Русия
e-mail: zablotskii@miigaik.ru

Светлана Василева
Природоматематическа гимназия „Иван Вазов“
бул. „Трети март“ 1
9300 Добрич, България
e-mail: svetlanaeli@abv.bg

GEODETIC TASKS IN PROGRAMMING TRAINING COURSES

Vladimir Zablotskii, Svetlana Vasileva

A typical training C ++ programming task for students and students studying cartography and surveying is presented. In the example, the flat rectangular coordinates of a search point are calculated by reverse geodetic detection with an additional checkpoint (the problem of Pothenot). In the program fragment the problem is solved by reducing it to the problem in the direct resection using the formulas of Delambre. The program calculates and outputs the values of the X, Y coordinates of the search point, if the discrepancy between the abscissa and the ordinates of the two solutions satisfies the rule for the "three-sigmas rule". The example uses procedural programming, making such tasks suitable for educational projects in Programming (C ++) classes for the "Mathematics" (Informatics) profile.
Key words: teaching C++ programming, computer program, Pothenot's problem, solution of an inverse geodetic angle resection with Delambre formulas.