УДК 378.147:[004.42+528]

C++ ДЛЯ СТУДЕНТОВ КАРТОГРАФОВ И ГЕОДЕЗИСТОВ: УЧЕБНАЯ ПРОГРАММА «ВЫЧИСЛЕНИЕ ПЛОСКИХ ПРЯМОУГОЛЬНЫХ КООРДИНАТ ТОЧКИ НА ТОПОГРАФИЧЕСКОЙ КАРТЕ»

^{1,2}Заблоцкий В.Р.¹Московский государственный технический университет имени Н.Э. Баумана, Москва;²Московский государственный университет геодезии и картографии, Москва,
e-mail: V.R.Zablotskii@yandex.ru

Разработана учебная программа для студентов картографов и геодезистов, изучающих основы программирования на языке C++. Программа вычисляет плоские прямоугольные координаты множества точек, заданных на листе топографической карты. Алгоритм решения задачи заключается в измерении длин перпендикуляров, опущенных от искомой точки на линии километровой сетки карты. Для вычисления координат используется юго-западный угол квадрата, в котором находится искомая точка. С целью получения более точных результатов определения координат в задаче также вычисляются невязки X и Y координат. Программа демонстрирует применение конструкции повторения – цикла while в варианте бесконечного цикла, остановка которого возможна только путем завершения программы. Несмотря на кажущуюся неопрятность такой конструкции, это позволяет пользователю экономить силы и время и не вводить дополнительные символы с клавиатуры для управления циклом. Также иллюстрируется условная конструкция в сокращенной форме записи и операция явного приведения типа данных. В результате расчета программа выводит на экран значения координат точек – абсциссу и ординату, с точностью до 1 метра. Разработанная программа иллюстрирует решение задачи вычисления плоских прямоугольных координат на основе применения технологии процедурного программирования.

Ключевые слова: обучение языку программирования C++, учебная геодезическая задача, вычисление плоских прямоугольных координат точек на карте

C ++ FOR CARTOGRAPHERS AND SURVEYORS: EDUCATIONAL PROGRAM «CALCULATION OF THE PLANE RECTANGULAR COORDINATES OF POINTS ON A TOPOGRAPHIC MAP»

^{1,2}Zablotskiy V.R.¹Bauman Moscow State Technical University, Moscow;²Moscow State University of Geodesy and Cartography, Moscow, e-mail: V.R.Zablotskii@yandex.ru

The training program for cartographers and surveyors studying the basics of programming in C++ was developed. The program calculates the rectangular coordinates of the set of points specified on the topographic map sheet. The algorithm for solving the problem consists in measuring the lengths of perpendiculars omitted from the desired point on the line of the kilometer grid of the map. The south-west corner of the square grid was used to calculate the coordinates were the desired point is located. The residuals of the X and Y coordinates also calculate in order to obtain more accurate results for determining the coordinates. This program demonstrates the use of the control statement for repetition – the while loop in the form of an infinite loop, which can only be stopped by terminating the program. Despite the apparent untidiness of this approach, it allows the user to save time and effort and not to enter additional characters from the keyboard for controlling program flow. Also the conditional construction in the abbreviated form of the record and the operation of the explicit reduction of the data types are illustrated. As a result of their use, the program displays the values of the coordinates of the points – abscissa and ordinate rounded to 1 meter. The developed programs illustrate the solution of the task of calculating the planned rectangular coordinates based on the application of procedural programming technology.

Keywords: teaching C ++ programming, training geodetic task, calculation plane rectangular coordinates of points on map

Несмотря на быстрое развитие науки о компьютерах, обучение программированию еще не прошло этап разделения по отраслям тематических знаний. Практически во всех высших учебных заведениях используется классическая схема преподавания, основанная на программировании задач, как правило, из области математики, одинаково пригодных для будущих инженеров разных специальностей. Однако в настоящее время процесс тематического разделения находится в стадии формирования, о чем косвенно свидетельствует появление, например учебника по

программированию Питера Готтшлинга «Современный C++ для программистов, инженеров и ученых» [1]. Очевидно, что нет недостатка в учебниках по программированию на языке C++ универсального типа, и среди них можно отметить классический, многостраничный учебник-справочник Стивена Прага, соавтор Дженеси Либерти [2–5]. Однако все эти учебники рассчитаны на широкий круг читателей, и поэтому в них отсутствуют примеры и задачи, ориентированные для профильных инженеров, например картографов и геодезистов.

Нашей целью является разработка набора типовых учебных геодезических программ и задач [4], которые могут использовать как преподаватели, так и студенты, обучающиеся по специальностям картографии и геодезии. Идет работа над сборником задач по программированию по С++ для студентов картографов и геодезистов. В него должны входить задачи геодезического содержания. Приведем несколько примеров таких задач. При изучении темы «Функции, их назначение и использование» уместной является следующая задача. *Напишите и запустите программу, которая вычисляет истинный азимут направления с помощью функции TrueAzimuth. Исходные данные – дирекционный угол в градусах и минутах и склонение меридианов в градусах и минутах задается в главной функции, и передаются в функцию TrueAzimuth в качестве параметров, которая вычисляет истинный азимут и выводит результат на экран.* Сформулируем еще одну задачу для изучения темы «Условные инструкции и переключатель». *Напишите и запустите программу с логической цепочкой if-else-if, которая вычисляет румб направления некоторой линии по ее дирекционному углу.* Очевидно, что программирование такого рода задач потребует специальных знаний из области геодезии и картографии. Такие задачи готовят студентов, будущих инженеров геодезистов и картографов, к решению своих специфических задач более эффективно, нежели общие типовые задачи. Автор выносит на обсуждение тезис о том, что разделение курсов программирования по специальностям, на основе создания специализированных лекций, учебников и сборников задач по программированию, есть задача актуальная и ее необходимо решать в настоящее время [6].

В данном сообщении рассматривается учебная программа для студентов картогра-

фов и геодезистов, изучающих основы программирования на языке С++, демонстрирующая вычисления плоских прямоугольных координат множества точек, находящихся на топографической карте.

Рассмотрим геодезическую постановку задачи. Пусть задан набор некоторых точек на топографической карте. Требуется вычислить плоские прямоугольные координаты этих точек. Как известно, прямоугольные X и Y координаты вычисляются по формулам

$$X_p = X_{ю} + \Delta X_{ю} \text{ и } Y_p = Y_3 + \Delta Y_3,$$

где $X_{ю}$ и $Y_{ю}$ значения координат юго-западного угла квадрата, в которой расположена заданная точка, $\Delta X_{ю}$ и ΔY_3 приращения координат заданной точки. Для получения более точных результатов определения прямоугольных координат точки в задаче вычисляются невязки X и Y координат. Поскольку суммы длин перпендикуляров $b_1 + b_2$ и $a_1 + a_2$, опущенных на противоположные стороны квадрата сетки должны быть равны 1000 м, т.е. длине стороны квадрата километровой сетки, находится разность между вычисленными значениями и истинными. Рассчитанные невязки распределяются между перпендикулярами с обратным знаком в пропорциях определяемых их длинами, например:

$$\Delta X_{ю} = \Delta X - \frac{b_1}{(b_1 + b_2)} \cdot erX,$$

$$\Delta Y_{ю} = \Delta Y - \frac{a_1}{(a_1 + a_2)} \cdot erY,$$

где ΔX и ΔY – приращения координат точки, erX , erY ошибки в графическом определении абсциссы и ординаты заданной точки.

Разработанная программа вычисляет плоские прямоугольные координаты заданных на карте точек, используя бесконечный цикл *while(1)* для многократного повторения вычислений координат точек.

```

01: #include <iostream>
02: using namespace std;
03:
04: int main (void)
05: {
06:     int MapScale;
07:     int XcoordinateOfPoint, YcoordinateOfPoint;
08:     int XcoordinateSouthWestCornerOfGrid, deltaX;
09:     int YcoordinateSouthWestCornerOfGrid, deltaY;
10:     double a1, a2, b1, b2, errorX, errorY;
11:     double incrementToX, incrementToY, decremtntToX, decremtntToY;
12:
13:     while(1) {
14:         cout <<"Введите знаменатель масштаба карты: ";
15:         cin >> MapScale;
16:         cout <<"Введите X Y координаты юго-западного угла сетки: ";

```

```

17:  cin  >> XcoordinateSouthWestCornerOfGrid
17:  >> YcoordinateSouthWestCornerOfGrid;
18:  cout <<"Введите расстояние от точки до западной и восточной"
18:  <<" линии рамки километровой сетки (в мм): ";
19:  cin  >> a1 >> a2;
20:  cout <<"Введите расстояние от точки до южной и северной"
20:  <<" линии рамки километровой сетки (в мм): ";
21:  cin  >> b1 >> b2;
22:
23:  errorX = (b1 + b2) * MapScale/1000 - 1000;
24:  errorY = (a1 + a2) * MapScale/1000 - 1000;
25:  cout <<"error X: "<< errorX <<"      error Y: "<< errorY << endl;
26:
27:  incrementToX = b1 * MapScale/1000 - errorX * b1/(b1 + b2);
28:  incrementToY = a1 * MapScale/1000 - errorY * a1/(a1 + a2);
29:
30:  deltaX = (int)incrementToX;
31:  deltaY = (int)incrementToY;
32:
33:  if((incrementToX - deltaX) >= 0.5)  deltaX++;
34:  if((incrementToY - deltaY) >= 0.5)  deltaY++;
35:
36:  XcoordinateOfPoint=XcoordinateSouthWestCornerOfGrid *1000 +deltaX;
37:  YcoordinateOfPoint=YcoordinateSouthWestCornerOfGrid *1000 +deltaY;
38:
39:  cout<<"Coordinates of Point: "<<"X = "<<XcoordinateOfPoint <<"m "
39:  <<" Y = " << YcoordinateOfPoint <<"m" << endl;
40:
41:  cout <<"Вводите новые данные для расчета: " << endl;
42:  }
43:  return 0;
44:  }

```

Рассмотрим код программы. В строках 06–08 объявляются переменные целого типа для знаменателя масштаба топографической карты *MapScale*, *X* и *Y* координат выбранной точки *XcoordinateOfPoint*, *YcoordinateOfPoint*, *X* и *Y* координат юго-западного угла километровой сетки *XcoordinateOfSWCorner*, *YcoordinateOfSWCorner*, а также приращений координат *deltaX*, *deltaY* искомой точки. В строках 09–11 объявляются переменные с плавающей точкой *a1*, *a2* для хранения значений длин перпендикуляров, опущенных из данной точки на западную и восточную сторону квадрата, а также длин перпендикуляров *b1*, *b2*, опущенных из данной точки на южную и северную сторону квадрата соответственно. Длины перпендикуляров выражаются в виде целых и десятых долей миллиметров. Переменные *errorX*, *errorY* используются для ошибок в графическом определении абсциссы и ординаты заданной точки. Переменные *incrementToX*, *incrementToY*, *decrementToX*, *decrementToY* содержат значения приращений – отрезков в метрах, на которые заданная точка отстоит от юго-западного и северо-восточного угла координатной сетки. Пользователь сначала вводит значение знаменатель масштаба карты, далее

значение *X* и *Y* координат юго-западного угла километровой сетки. Отметим, что от данного угла координатной сетки и выполняется расчет координат искомой точки. Затем пользователь вводит значения длин перпендикуляров, опущенных из данной точки на западную и восточную сторону квадрата, а также длин перпендикуляров, опущенных из данной точки на южную и северную сторону квадрата. В строках 23–24 вычисляются невязки координат по *X* и *Y*. Полученные результаты выводятся на экран. Затем в строках 27–28 вычисляются приращения абсциссы и ординаты точки по формулам, указанным ранее. Далее в строках 30 и 31 переменные *deltaX*, *deltaY* получают приращения координат с точностью до целых метров. При этом используется операция приведения типов, позволяющая взять целое число значений координат (*int*) *incrementToX*, (*int*) *incrementToY*. Однако при отбрасывании дробной части могут возникать нежелательно большие погрешности, поэтому для устранения погрешностей больших 0,5 м используются условные конструкции вида: *if((incrementToX - deltaX) >= 0,5) deltaX++*. Если полученная разность превышает 0,5, то приращение координат увеличивается на 1. Аналогичная ин-

струкция выполняется и для *incrementToY*. В строках 36 и 37 вычисляются координаты искомой точки *XcoordinateOfPoint*, *YcoordinateOfPoint*, для этого координаты юго-западного угла сетки преобразуются в метры умножением на 1000 и добавляется приращение координат. В строке 39 на экран выводится результат вычислений.

Предположим, что пользователь ввел следующие данные: карта масштаба 1:25000, координаты юго-западного угла координатной сетки $X = 6065$, $Y = 4311$, длины перпендикуляров (в миллиметрах) соответственно равны $a1 = 11,4$, $a2 = 28$, $b1 = 23,6$, $b2 = 16$. В результате программа напечатает на экране:

Погрешность X: -10 Погрешность Y: -15
 Координаты точки: X = 6065596 м Y = 4311289 м
 Вводите новые данные

Отметим особенность данной программы, которая заключается в применении бесконечного цикла *while(1)* для многократного повторения вычислений прямоугольных координат точек. Данный пример демонстрирует, как использовать бесконечный цикл, который в данном случае является полезным инструментом, сокращающим диалог пользователя с программой при ее выполнении. Такой подход имеет определенное преимущество для пользователя, которому не надо каждый раз после выполнения очередного шага цикла вводить дополнительный символ, означающий, что цикл должен быть продолжен или, наоборот, остановлен. Возможны разные варианты создания бесконечных циклов, например с помощью конструкций

циклов *for*, *while* и *do-while*. При создании бесконечного цикла *while* следует поместить в круглые скобки условия, следующего за ключевым словом *while* ненулевую константу, например число 1, которая будет означать «истину» и цикл никогда не закончится. Такая конструкция цикла подходит для случая, когда программа вычисляет координаты некоторого набора точек по карте и не выполняет дальнейшей обработки полученных координат. По сути дела вычисление координат является основной и заключительной задачей программы. В случае, если такое завершение цикла в программе неприемлемо, можно использовать другой вариант цикла, управляемого изнутри, как например это показано в следующем фрагменте кода:

```
do
{
. . . . . // полезная нагрузка тела цикла
. . . . .
cout << "Продолжить программу? Введите (Y/y): ";
cin >> noneStopCharacter;
}while((noneStopCharacter=='Y') || (noneStopCharacter=='y'));
```

Такой цикл будет многократно выполняться, если пользователь на вопрос «Продолжить программу? Введите (Y/y):», будет вводить букву Y или y. В результате условие $((noneStopCharacter=='Y') || (noneStopCharacter=='y'))$ получает значение «истина» и цикл продолжается. Очевидное преимущество бесконечного цикла над последним логически более корректным, состоит в отсутствии необходимости ввода букв «Y» или «y» со стороны пользователя.

$errorX * b1 / (b1 + b2) + b2 * MapScale / 1000 - errorX * b2 / (b1 + b2) = 1000$ для координаты X и $a1 * MapScale / 1000 - errorX * a1 / (a1 + a2) + a2 * MapScale / 1000 - errorX * a2 / (a1 + a2) = 1000$ для координаты Y. Нельзя сказать, что теперь при выполнении расчетов на компьютере такие проверки полностью бесполезны. Конечно же, в своих расчетах компьютер не ошибается в отличие от человека. Человек может совершить ошибку в расчетах, компьютер – нет. Однако дополнительная проверка может оказаться полезной при контроле входных данных. Поэтому такого рода проверки можно включать в код программ, полностью следуя алгоритмам обработки исходных данных, применяемым в геодезии. Эти алгоритмы были разработаны именно с уче-

Еще одно замечание. В решении этой и других задач в геодезической практике часто выполняется контроль вычислений. В рассмотренном случае, например, это контроль координаты X и Y, который выполняется по формулам: $b1 * MapScale / 1000 -$

том «человеческого фактора» и если они переносятся в компьютерные программы, то должны быть веские причины для этого.

Целесообразным в решении данного вопроса является следующий подход. Если в программе переменные инициализируются и отсутствует ввод данных с клавиатуры, то в программе можно не выполнять контроль вычислений. В этом случае возможно написать код программы, конкретно под фиксированные переменные. Если же планируется использовать ввод данных в переменные с клавиатуры, контроль вычислений позволит обнаружить ошибки ввода. В этом случае надо предусмотреть возможные разные значения геодезических величин, чтобы программа могла их корректно обрабатывать. Обычно это усложняет логику программы, приводит к программам с большим объемом кода, однако же делает программу более универсальной.

Выводы

Разработана учебная программа для студентов картографов и геодезистов, изучающих основы программирования на языке C++ в геодезическом вузе. Программа демонстрирует вычисление плоских прямоугольных координат некоторого множества точек, заданных на топографической кар-

те. Подробно описывается код программы и поясняется назначение использованных инструкций. Среди них находятся конструкция повторения или цикл *while*, условная конструкция *if*, операция явного приведения типов данных (*int*), позволяющие округлить полученные значения координат до целых метров. Разработанная программа иллюстрирует решение задачи вычисления плоских прямоугольных координат на основе применения технологии процедурного программирования.

Список литературы

1. Готтшлинг П. Современный C++ для программистов, инженеров и ученых. – М.: ООО «И.Д. Вильямс», 2016. – 512 с.
2. Прата С. Язык программирования C++. Лекции и упражнения. 6-е изд. – М.: ООО «И.Д. Вильямс», 2014. – 1248 с.
3. Либерти Дж. Освой самостоятельно C++. 10 минут на урок. 2-е изд. – М.: ООО «И.Д. Вильямс», 2004. – 352 с.
4. Либерти Дж. Освой самостоятельно C++ за 24 часа. 6-е изд. – М.: ООО «И.Д. Вильямс», 2017. – 448 с.
5. Либерти Дж. Освой самостоятельно C++ за 21 день. 3-е изд. – М.: ООО «И.Д. Вильямс», 2001. – 816 с.
6. Zablotskii V.R. Teaching C++ programming language at Moscow State University of Geodesy and Cartography through the geodesic problems and programs. Proc. 17th International Multidisciplinary Scientific Geoconference SGEM. 2017, Vol. 17, Informatics, Geoinformatics and Remote Sensing, Issue 21, P. 641–646, 29 June – 5 July, 2017, Albena, Bulgaria.