УДК 616-092.9:612.438:577.118

C++ ДЛЯ СТУДЕНТОВ КАРТОГРАФОВ И ГЕОДЕЗИСТОВ: УЧЕБНАЯ ПРОГРАММА «ВЫЧИСЛЕНИЕ ГЕОГРАФИЧЕСКОГО АЗИМУТА НАПРАВЛЕНИЯ»

Заблоцкий В.Р.

Московский 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 С++. Программа вычисляет географический азимут направления некоторой линии, заданной дирекционным углом и известным средним зональным сближением меридианов для листа топографической карты. Программа демонстрирует объявление переменных, применение условных конструкций в полной и сокращенной формах, операцию явного приведения типов данных, а также преобразование углов из градусного и минутного представления в градусное с целой и дробной частью и обратно. Выполнен компьютерный эксперимент с целью модификации кода программы в логически более ясную форму записи, рассмотрены преимущества и недостатки модифицированной версии программы. Разработанные программы иллюстрируют решение задачи вычисления географического азимута на основе применения технологии процедурного программирования.

Ключевые слова: обучение языку программирования С++, учебная геодезическая задача, расчет географического азимута направления

C ++ FOR CARTOGRAPHERS AND SURVEYORS: EDUCATIONAL PROGRAM «CALCULATION OF THE TRUE AZIMUTH OF A LINE DERICTION»

Zablotskiy V.R.

Moscow 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 geographic azimuth of a line drawn on a topographic map of on base the grid angle and the average convergence of meridians for the map sheet. The program demonstrates the declaration of variables the using of conditional constructions in full and abbreviated forms then operation of explicit conversion of data types and at last the conversion of angles from degrees and minutes to degrees with integer and fractional parts as forward and backward. A computer experiment was performed to modify the program code to a logically more clear form of the program. The advantages and disadvantages of the modified version of the program are considered. These programs illustrate the solution of the task of calculating geographic azimuth based on the application of procedural programming technology.

Keywords: teaching C ++ programming, training geodetic task, calculation geographic azimuth

Рассматривается учебная программа для студентов картографов и геодезистов, изучающих основы программирования на языке С++. В настоящее время имеется много хороших книг по программированию на языке С++, например классический, многостраничный справочник Стивена Прата, небольшой по объему самоучитель Джесси Либерти или учебник, иллюстрированный комиксами, Стефана Дениса [1–3]. Однако отсутствуют учебники, ориентированные на подготовку профильных инженеров, например картографов и геодезистов. Нашей целью является разработка набора типовых учебных геодезических программ [4], которые могут использовать как преподаватели, так и студенты, обучающиеся по специальностям картографии и геодезии. Задачей данной работы является разработка программы, демонстрирующей вычисления географического (истинного) азимута направления.

Рассмотрим содержательную геодезическую постановку задачи. Пусть задан дирекционный угол α некоторой линии на топографической карте. Известно также среднее значение зонального сближения меридианов γ для данного листа карты. Требуется вычислить географический азимут направления этой линии. Как известно, географический азимут направления вычисляется по формуле: $A_{ист} = \alpha + \gamma$. Для точек, лежащих к востоку от осевого меридиана, склонение меридианов считается положительным, для точек, лежащих к западу от осевого меридиана – отрицательным. В задачах подобного типа обычно полагается, что углы заданы в градусном и минутном представлении с точностью до 1 минуты, например, если дирекционный угол и сближение меридианов соответственно равны: $\alpha = 1°10'$, $\gamma = 2°05'$, географический азимут направления $A_{ист} = 359°05'$.

Разработанная программа вычисляет географический азимут направления, используя преобразование данных углов из градусной и минутной меры в формат градусы

с целой и дробной частью. Такой прием позволяет вычислять сумму или разность углов в одно арифметическое действие, используя операторы «плюс» или «минус». Однако на практике чаще используют градусно-минутную угловую меру, при которой требуется выполнить два арифметических действия, одно для значения градусов, другое для минут. Использование формата градусы с целой и дробной частью упрощает логику программы и делает ее более понятной.

```
01: #include <iostream>
02: using namespace std;
03:
04: int main(void)
05: {
06: double degrees, minutes, gridAzimuth, zonalConvergenceOfMeridians;
07:
08: cout <<"Введите дирекционный угол линии (градусы,пробел,минуты):";
09: cin >> degrees >> minutes;
10: gridAzimuth = degrees + minutes/60;
11:
12: cout <<"Введите зональное сближение меридианов "
13:      <<" (градусы,пробел,минуты): ";
14: cin >> degrees >> minutes;
15:
16: if(degrees >= 0)
17:      zonalConvergenceOfMeridians = degrees + minutes/60;
18: else
19:      zonalConvergenceOfMeridians = degrees - minutes/60;
20:
21: double TrueAzimuth = gridAzimuth + zonalConvergenceOfMeridians;
22:
23: if(TrueAzimuth > 360) TrueAzimuth = TrueAzimuth - 360;
24:
25: if(TrueAzimuth < 0 ) TrueAzimuth = TrueAzimuth + 360;
26:
27: degrees = (int)TrueAzimuth;
28:
29: minutes = (TrueAzimuth - degrees)* 60;
30:
31: cout <<"Географический азимут направления: " << degrees <<"°"
31:      << minutes <<"'" <<endl;
32:
33: return 0;
34: }
```

Рассмотрим код программы. В строке 06 объявляются переменные с плавающей точкой двойной точности: *degrees, minutes, gridAzimuth, zonalConvergenceOfMeridians* для хранения значений градусов, минут, дирекционного угла и зонального сближения меридиан с дробной частью. Переменные *degrees*, *minutes* используются для ввода данных значений с клавиатуры – сначала для значения дирекционного угла направления, а затем для значения зонального сближения меридиан. Обращаем внимание на многократное использование переменных *degrees* и *minutes*. В начале программы они используются для временного хранения одноименных величин дирекционного угла (строки 09–10). Затем в строках 14–19 эти переменные сохраняют одноименные величины для сближения меридианов. В конце программы (строки 27–33) переменные используются для хранения значений градусов и минут географического азимута направления линии. Такое многократное использование переменных – часто встречающийся прием в программировании, следует лишь использовать переменные соответственно их смысловым именам. Пользователь сначала вводит значение дирекционного угла в градусах и минутах. Затем пользователь вводит значение сближения меридианов, тоже в градусах и минутах. Переменные *gridAzimuth* и *zonalConvergenceOfMeridians* начинают использоваться в строках 10 и 17 (19) для хранения углов в формате градусов с дробной частью. Здесь требуется пояснение назначение условной конструкции *if-else* представленной в строках 16–19. Дело в том, что сближение мериди-

анов может иметь как положительное, так и отрицательное значение, при этом в геодезической практике знак минус ставится только перед градусами, например запись γ = 2 °05' означает, что и минуты тоже надо рассматривать со знаком минус. Тем не менее, с клавиатуры вводятся раздельно два числа, одно со знаком «+» для значения градусов, другое со знаком «–» для значения минут. В этой связи значение переменной *zonalConvergenceOfMeridians* для положительных значений вычисляется как *zonalConvergenceOfMeridians = degrees + minutes/60* , а для отрицательных значений вычисляется с помощью инструкции *zonalConvergenceOfMeridians = degrees – minutes/60*. В строке 21 вычисляется географический азимут направления по формуле $A_{ист} = \alpha + \gamma$.

TrueAzimuth = gridAzimuth +
+ zonalConvergenceOfMeridians.

В строках 23 и 24 иллюстрируется применение сокращенной формы условной конструкции, *if* без ветви *else*, в отличие от строк 16–19, где использовалась такая же условная конструкция, только в полной форме. По определению географический азимут направления – это угол, лежащий в диапазоне от 0 ° до 360 °. Поэтому если в результате расчета азимут превышает 360 °, то из полученного значения вычитается полный круг, иначе *TrueAzimuth = TrueAzimuth – 360*, если азимут получился отрицательным, то к полученному значению добавляется полный круг *TrueAzimuth = TrueAzimuth + 360*.

Далее, инструкции, находящиеся в строках 27 и 29, предназначены для преобразования значения азимута направления в формат градусов и минут с целью вывода этих значений на экран. В строке 27 используется операция приведения типов, позволяющая взять целое число градусов *(int)TrueAzimuth* и хотя далее это число присваивается переменной *degrees* с плавающей точкой двойной точности, т.е. добавляется дробная часть, состоящая из нулей, при выводе на экран нулевая дробная часть не печатается. В строке 29 вычисляется значение минут географического азимута, для этого из переменной *TrueAzimuth*, содержащей значения угла в градусах с дробной частью, вычитается значение переменной *degrees,* которая содержит только целую часть градусов. Полученный остаток умножается на 60 для перевода этой величины в угловые минуты. Затем в строке 31 на экран выводится результат расчета географического азимута направления.

Предположим, что пользователем были введены следующие данные, дирекционный угол направления линии и зональное сближение меридианов соответственно равны: α = 359 °20′, γ = –1 °30′. В результате программа напечатает на экране: «Географический азимут направления: 357 °50’».

В заключение рассмотрим другую версию программы, представленную далее, в которой используются переменные *degrees, minutes* целого типа. Этот тип переменных более соответствует замыслу использовать углы в градусном и минутном выражении с точностью до 1 минуты. Однако во избежание целочисленного деления при переводе углов в формат с целой и дробной частями приходится использовать явное преобразование типов. Как видно в строках 11, 18 и 20 используется *(float)* преобразование в вещественный тип. В случае отсутствия такого преобразования деление выполнялось бы как целочисленное, что приводило бы к потере угловых минут.

```
01: #include <iostream>
02: using namespace std;
03:
04: int main(void)
05: {
06: int degrees, minutes;
07: double gridAzimuth, zonalConvergenceOfMeridians, TrueAzimuth;
08:
09: cout <<"Введите дирекционный угол линии (градусы,пробел,минуты):";
10: cin >> degrees >> minutes;
11: gridAzimuth = degrees + (float)minutes/60;
12:
13: cout <<"Введите зональное сближение меридианов "
14:      <<" (градусы,пробел,минуты): ";
15: cin >> degrees >> minutes;
16:
17: if(degrees >= 0)
18:      zonalConvergenceOfMeridians = degrees + (float)minutes/60;
19: else
20:      zonalConvergenceOfMeridians = degrees - (float)minutes/60;
21:
```

```
22: TrueAzimuth = gridAzimuth + zonalConvergenceOfMeridians;
23:
24: if(TrueAzimuth > 360) TrueAzimuth = TrueAzimuth – 360;
25:
26: if(TrueAzimuth < 0 ) TrueAzimuth = TrueAzimuth + 360;
27:
28: degrees = TrueAzimuth;
29:
30: minutes = floor((TrueAzimuth – degrees)* 60 + 0.5);
31:
32: cout <<»Географический азимут направления: « << degrees <<»°»
32: << minutes <<»'» <<endl;
33:
34: return 0;
35: }
```

Другим преимуществом использования целочисленных типов для переменных *degrees, minutes* является возможность использования неявного преобразования типов, что происходит в строке 28. Выражение вида *degrees = TrueAzimuth* присваивает переменной целого типа *degrees* значение переменной с плавающей точкой *TrueAzimuth*, результатом такого присваивания будет то, что переменная *degrees* получит только целую часть градусов от переменной *TrueAzimuth*, при этом дробная часть отбрасывается. Это как раз и требуется. Однако имеется и некоторое усложнение инструкций кода, требуемое для того, чтобы вывести на экран значение угла в минутах без потери точности. Для этой цели используется функция округления *floor*. Функция *floor* выполняет округление вещественного числа до целого числа с недостатком, например число 3,123 будет округлено до 3. В стандартной библиотеке математических функций C/C++ имеется и другая функция *ceil*, выполняющее округление вещественного числа до целого с избытком, например число 3.123 будет округлено до 4. В программе используется функция *floor*, хотя можно с таким же успехом использовать *ceil*. Инструкция в строке 30 позволяет вывести значение минут без потери точности:

minutes = floor((TrueAzimuth – degrees) 60 + 0,5).*

Поскольку переменная *minutes* является целого типа, то присваивание без использования функции *floor* приводит к отбрасыванию дробной части, поэтому вначале округляемое значение увеличивается на 0,5, а затем применяется функция округления.

Выводы

Разработана учебная программа для студентов картографов и геодезистов, изучающих основы программирования на языке C++ в геодезическом вузе. Программа демонстрирует вычисления географического азимута направления некоторой линии, заданной на топографической карте. Подробно описывается код программы и поясняется назначение использованных инструкций. Среди них находятся условные конструкции в полной *if-else* и сокращенной формах *if*, операции явного приведения типов данных *(int)*, преобразования углов из формата градусы и минуты в формат градусы с целой и дробной частью и обратно. Также выполнен компьютерный эксперимент с целью модификации кода программы, рассмотрены преимущества и недостатки модифицированной версии программы. Разработанные программы иллюстрирует решение задачи вычисления географического азимута на основе применения технологии процедурного программирования.

Список литературы

1. Прата С. Язык программирования C++. Лекции и упражнения. 6-е изд. – М.: ООО «И.Д. Вильямс», 2014. – 1248 с.

2. Либерти Дж. Освой самостоятельно C++. 10 минут на урок. 2-е изд. – М.: ООО «И.Д. Вильямс», 2004. – 352 с.

3. Денис С.Р. C++ для чайников. 7-е изд. – М.: ООО «И.Д. Вильямс», 2016. – 400 с.

4. 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.