A New Approach to Teaching C++ Programming for Cartography Students by Means Training Programs with Emphasizing Cartography and Geodesy

Vladimir Zablotskii [a, b]

[a] *Bauman Moscow State Technical University,*
[b] *Moscow State University of Geodesy and Cartography, V.R.Zablotskii@Yandex.ru*

Abstract: A new course of C++ programming for cartographers and surveyors has been developed. Pedagogical experiments were carried out in the period of 2009–2019 in Moscow State University of Geodesy and Cartography. The new course focuses on the use of cartographic tasks and geodetic exercises to illustrate various programming language constructions. Direct and inverse geodetic problem, position determination via a topographic map, work with a theodolite when performing angular measurements, leveling, etc. are considered in training modules. Programs are used as supporting data during the lectures, and as tasks for practical. This is the main purpose of the training. Currently, more than fifty training programs are used in the training process. All programs contain no more than 60 lines of code to ease the understanding of the program by students. The examples the typical training programs for students studying the C++ are discussed. The first program Theodolite is designed to study the concept of class, constructor, destructor, and object-oriented programming in general. The second program focuses to solve the inverse of the geodetic task.

Keywords: Teaching C++ programming, Training computer programs, Cartographic and geodetic tasks

1. Introduction

Currently teaching of programming in higher educational institutions of Russia is still carried out according to the traditional scheme based on the fact that the study of programming is performed on tasks in the field of higher mathematics and logic equally suitable for future engineers of different specialties. Although, the reality has changed dramatically over the past years new programming languages such as C++, C#, Java appeared. Nowadays they are being studied in universities of the country. In fairness it must be said that the assembler language is being studied in a less degree, while the high–level programming languages are emphasized in the teaching process. However, almost all higher education institutions use the conventional teaching scheme without dividing of educational process by different specialties for future engineers. The subject-related division of programming courses in the fields of engineering is still in the process. This is a worldwide trend, as evidenced by the appearance of a number of textbooks on programming, designed for engineers and scientists (Bronson 2013, Gottschling 2016). Foundational textbooks (Prata 2011, Liberty, Cashman 2002, Liberty, Cadenhead 2011, Schildt, 2003, Stroustrup, 1997) are designed for a wide range of readers and therefore there are no examples of programs and tasks for specific engineering specialties, for example surveyors. Currently there is no C++ programming textbook for students of both geodesy and cartography. In addition there are few C++ programs available in the Internet which might be helpful for teaching of the first-year students dealing with general geodesy.

The C++ programming course for undergraduate students has been developed at the Moscow State University of Geodesy and Cartography (MIIGAiK). This course is based on geodetic training tasks and intended to meet the needs of surveying industry where the university plays one of the leading roles. The lectures of this course were illustrated with tasks of general geodesy. Practical lessons at the computer classes and homework are to prepare students to use knowledge of C++ programming methods to solve various engineering and surveying problems.

Our aim is to develop a set of sample educational geodetic programs and tasks (Zablotskii 2009) designed to teach C++ programming students making a specialty of geodesy and cartography. Programming of geodetic tasks requires students to have some specific knowledge in geodesy and cartography as well as in programming. Therefore, the programming of geodetic tasks unlike solving typical mathematical problems might train future surveyors and cartographers to solve their specific practical problems. The author argues that effective teaching of programming in the field of engineering should be based on subject–related lectures, textbooks and collections of programming tasks. This task is very urgent at the present time, and it is to be solved promptly with a view to improving the quality of teaching programming in higher education.

As compared to the conventional training model the approach proposed has the following advantages. First, students learn programming that aims at creating programs in the field of cartography and geodesy. They become familiar with the real circumstances that they may be challenged in their future. Secondly, students solve algorithms for a number of widespread geodetic and cartographic tasks. They gain experience in developing C++ programs to solve such thematic tasks. Thirdly, students develop skills of using computer technology not only with ready-made software, but they are also guided to create new programs to solve their own problems. This approach contributes not only to the study of programming, but also to the study of cartography and geodesy, since students analyze the issues of cartography and geodesy during practical the context of C++ programming.

Modern devices of the geodetic industry are programmable devices and hardware systems. It will certainly help to use GIS and digital maps more effective with basic knowledge of computer science and fundamentals of programming. The author-developed C++ programming course for MIIGAiK students takes into account those needs and is based on cartographic and geodetic training tasks.

Today there are written about fifty training programs on different topics of C++ programming course. These are some examples of geodetic training programs used in course.

Title of the program	*Goal of the program*
Welcome to MIIGAiK	introduction to the process of creation of a code of a program, compilation and debugging
Number Pi	output to the console of the number π with a different number of significant digits is studied
Average convergence of meridians	instructions of console output for different symbols and their composition are studied
Radius of Earth orbit	the error of memory overflow is studied
Height of a point on the map	the types of variables and arithmetic operators are studied
Column and zone number	the ternary operator (?:) is studied
Collimation error	if-else instructions in full and short form for computing the angular error of a theodolite's telescope are studied
Rhumb (reduced bearing)	the "logical chain" formed by nested if instructions is studied
Denominator of map scale	the switch instruction on the example of computing the horizontal equivalent is studied
Converting an angle from radian to degrees	the instruction of a for–loop is studied
Rectangular coordinates	the instruction of the infinite while loop on the example of calculating the coordinates of points is studied
Gaussian convergence of meridians	the convergence of meridians using while loop for a point given by the latitude and longitude is studied
Slope of the line	the instructions of the do–while loop on the example of calculating the slope of a line on a topographic map are studied
Calculating the angle	the declaration and calling the simple function which calculating degrees, minutes and seconds are studied
Direct angular tying	the trigonometric functions of the standard library and user's cotangent function are studied
Inverse Geodetic problems	the trigonometric functions of double precision and prototype of user' function are studied
Change the back azimuth	the pointer variable which change the back azimuth in main function is studied
Zero offset of circle	the reference which pass the readings to function for computing ZO of vertical circle of theodolite is studied
Theodolite	the multi-modular program using the measuring angles by the method of full sets is studied
Topographic map	the structures and technique for calculating the declination of the magnetic needle on the old map are studied
Theodolite traverse	usage of an arrays which store the angles of theodolite traverse is studied
Overload function	the overload function on the example of converting the angle from degrees to radians is studied
Adrianov's compass	the class and access to class members from functions are studied
Levelling rod	studying the constructor and destructor of a simple class
Geneva ruler	simple inheritance are studied
Bussol	multiple inheritance is studied
Triangle and Angle	friendly classes using the example of the friends classes "Angle" and "Triangle" are studied

The development and testing of training computer programs for geodesy and cartography students allows formulating a number of requirements for these programs. The following are the main ones. The training computer program must be comprehensible; therefore it must consist of a small amount of code. The 50–60 lines on average are enough. Each training task might be devoted to one or at most two programming language constructs. Programs are to illustrate all major language constructs. It is desirable that the code of the training computer program is written in a modern programming style and is read and understood easily. It will be achieved if the names of program objects are common and semantic names, and are understood by geodesy and cartography students. Examples of such names might be *declinationOfMagneticNeedle, convergenceOfMeridians*, etc. Each program should be attended by a specific manual.

2. Method and Materials

The work has been carried out via a personal computer (PC) with Microsoft Windows 8.1 operating system. In this work was used the free open source Code::Blocks environment and GCC C++ compiler. The Code::Blocks/gcc package generates 32-bit programs. The program in this paper runs within a command line prompt and writes out to the command line. The next object-oriented program Theodolite was developed for educational purpose.

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

class Theodolite {
  private:
    char model [7];
    int number;
    int accuracyOfMeasurement;
  public:
    Theodolite(char *model, int
                number, int accuracy);
    Theodolite(int number);
    ~Theodolite(void);
    void ShowSpecifications(void);
};

Theodolite::Theodolite(char *model,
           int number, int accuracy)
{
 strncpy(Theodolite::model, model,7);
 Theodolite::number = number;
 accuracyOfMeasurement = accuracy;
}
Theodolite::Theodolite(int number)
{
 strncpy(Theodolite::model,"2T30",7);
 Theodolite::number = number;
 accuracyOfMeasurement = 30;
}
Theodolite::~Theodolite(void)
{
 cout <<"myTheodolite is destroyed."
      <<" Memory is free." << endl;
}
void Theodolite::
          ShowSpecifications(void)
{
 cout <<"Theodolite model: "<< model
      << ", " <<" device number: "
      << number << endl;
 cout <<"Angle measurement accuracy:"
      << accuracyOfMeasurement
      <<" seconds" << endl;
 };
 void CreateMyTheodolite()
 {
  Theodolite myTheodolite(56789);
  myTheodolite.ShowSpecifications();
 };
int main(void)
{
 Theodolite theodolite("3T5KP",
                         12345, 5);

 theodolite.ShowSpecifications();
 CreateMyTheodolite();

 return 0;
}
```

This is a typical tutorial program through which students become familiar with the concept of class, constructor, destructor, and object-oriented programming in general.

The second training program intended at solving an inverse geodetic problem was developed. The following facts are included into practice for students as a reminder of the inverse geodetic task. Suppose the plane rectangular coordinates of the initial points are A (X_A,Y_A) and B (X_B,Y_B). It is required to determine the length of the horizontal distance of the line AB and the grid bearing of the line direction. As far as is known, in order to solve the inverse geodetic problem, the following formula $tg(r_{AB}) = \frac{\Delta Y}{\Delta X} = \frac{Y_B - Y_A}{X_B - X_A}$ is used. The increments of the coordinates of the B point relative to the point A are calculated with according to the formulae: $\Delta Y = Y_B - Y_A$ and $\Delta X = X_B - X_A$. Further, the length of the horizontal distance of the AB line is found via the rule of Pythagoras: $d_{AB} = \sqrt{\Delta X^2 + \Delta Y^2}$. To convert a geodesic bearing angle to a grid bearing, the known formulae are used.

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
void convertingRadianToDegMinSec(
 long double radian,int&,int&,int&);

int main (void)
{
 int degrees,minutes,seconds;
 double  length;
 long double xA,yA,xB,yB,dX,dY,angle;
12:
13:   cout<<"Geodetic coordinate system is"
13:       <<" used"<< endl;
14:   cout<<"Enter X and Y coordinates of "
14:       <<" point A: ";
15:   cin  >> xA >> yA;
16:   cout<<"Enter X and Y coordinates of "
16:       <<" point B: ";
17:   cin  >> xB >> yB;
18:
19:   length = sqrt( pow(dX,2)+pow(dY,2));
20:
21:   cout<<"Distance between two points: "
21:       << setiosflags(ios::fixed)
21:       << setprecision(2)<< length
21:       << " m" << endl;
22:
23:   angle = atan2l(dY,dX);
24:   if(angle < 0) angle = 2*M_PI + angle;
25:
26:  convertingRadianToDegMinSec(angle,
26:               degrees,minutes,seconds);
27:
28:   cout<<"Grid bearing of a line AB: "
28:   <<degrees<<"° "<<minutes <<"\' "
28:   <<seconds <<"\""<< endl;
29:
30:   return 0;
31:  }
32:  void convertingRadianToDegMinSec(long
32:  double radian,int& degrees,int&
32:                   minutes,int& seconds)
33:  {
34:   long double degreeWithFractPart =
34:                         radian*180/M_PI;
35:   degrees = (int)degreesWithFractPart;
36:
37:   long double minutesWithFractPart =
37:     (degreesWithFractPart-degrees)*60;
38:   minutes = (int)minutesWithFractPart;
39:
40:   long double secondsWithFractPart =
40:    (minutesWithFractPart - minutes)*60;
41:   seconds = (int)secondsWithFractPart;
42:
43:   if(secondsWithFractPart-seconds >=
43:                                   0.5)
44:   seconds++;
45:   if(seconds == 60) {
46:     seconds = 0;
47:     minutes++;
48:   }
49:   if(minutes == 60) {
50:     minutes = 0;
51:     degrees++;
52:   }
53:  }
```

3. Discussions

3.1 Analysis of first source code

The first program has the *Theodolite* class including the two constructors, destructor and method for visualization of technical parameters of theodolite. The *Theodolite* class definition (lines 06–16) contains a member function called *ShowSpecifications* (lines 34–38) that displays a model of theodolite, number of device and accuracy of angle measurement on the screen (lines 36-37). Lines 12-23 define the first constructor for *Theodolite* class and lines 24-29 define the second constructor. The constructor has the same name as its *Theodolite* class. The constructor has not a return type, because constructors cannot return values. These facts should be communicated to students and known to them at the beginning of the study of the topic "Classes and Objects". When a new object is creating, parameters in the parentheses goes to the object. Line 14 indicates that first class *Theodolite's* constructor has a character array parameter called *model* also two *int* parameters, one is *number* and other is *accuracy*. The second constructor of the *Theodolite* class has only one parameter, this is a number of device (lines 24-29).

The lines 30-33 define the destructor for class *Theodolite*. The name of the destructor for a class is the tilde character (~) followed by the class name. A class's destructor is called implicitly when an object is destroyed. This occurs when program execution leaves the scope in which that object was instantiated. This is the next important information for students studying classes and objects. For this purpose we create the object *myTheodolite* in line 41 inside the function called *CreateMyTheodolite*.

The program demonstrates the order in which constructors and destructor are called for objects of class Theodolite containing the various values of data members. Each object of class Theodolite contains an integer (number) and a character array (model) that are used in the program's output to identify the object.

The training process usually involves the launch of the compiled program on the PC and the subsequent analysis of the program instructions. An interactive electronic board is used. Application of an interactive board technology allows a teacher to concentrate on analyzing issues of programming. Conventional training method suggested usage of white board was not effective and put some extra load on the teacher. Moreover it has become difficult to work with a modular program which is based on a code consisted of more than one hundred instructions. Application of such a technology increases the efficiency of the teaching process and allows reducing the load on the teacher significantly. Even using an interactive whiteboard as a display device, improve the efficiency and intensiveness of practical, makes them interesting for students. Code analysis with visualization of control flow when calling functions and returning values is usually performed by the student. As programming skills develop, students have the task to analyze the program code orally. Naturally, the initial programs contain only a few lines. Then the number of lines in a code curriculum increases. Oral code analysis involves explaining each line of code and answering the questions what was its purpose and what the given instruction line does.

Sometimes it is difficult for a beginner to explain how the program written on the C ++ language works. In such cases, at first answers like: "I see …" are acceptable. For example, in line 06 of the program above, it might be heard the following: "I see the class with the name *Theodolite*. This class has private and public members and in public section there are two constructors and destructor". After the learner begins to recognize the language constructs, he should ask questions what is the aim of those constructs of the program.

The peculiarity of teaching a programming course with a focus on cartography and geodesy is that it is conducted on junior courses, and at the same time with the course of general geodesy. Therefore, the sequence of the training computer programs must be in the fairway of course of general geodesy. It is obvious that the use of programs containing geodetic tasks and exercises unknown for students is ineffective. Students have to learn them in a basic geodesy course first. Another feature of the training program development is that the programming language issues should be studied in a certain sequence. This inevitably leads to the fact that in the initial stages of training the teacher can operate a very narrow range of language constructs. As a result, some geodetic training tasks are solved, but to the point of view of a specialist, least effectively. For example, in the program illustrating loops, it is not preferable to use functions, in particular rounding ones, if the concept “function” is considered after the concept “loops”.

In next part a training program intended at solving an inverse geodetic problem is described. Then the explanation of what the program does and how it does it is given. This is an example of a guide for the program. The text is put into a methodological guide for students learning the C++ programming language.

3.2 Analysis of second source code

The training program solves the inverse geodetic problem using the standard mathematical function for arctangent calculations of the angle, and the user-defined transformation function so the angle is measured in degrees and not in radians. To calculate the value of the arctangent with extended precision the *atan2l* function is used. This function calculates the arctangent in the range $-\pi$ to π. Two arguments are passed to the *atan2l* function. Here they are increments along the X and Y coordinates, for example *atan2 (ΔY, ΔX)*. Since the grid angles in geodesy are expressed of positive values, and are measured from the north direction, a geodetic coordinate system is implemented in the program.

The program consists of two functions. A user enters all the necessary data for the calculation to the main function. This are the coordinates of point A (X_A, Y_A) and the coordinates of point B (X_B, Y_B). Then the coordinate increments are calculated, and the results of calculations are displayed on the screen as a control element. The horizontal distance is calculated by the Pythagorean formula. Here the standard library mathematical function of the square root (*sqrt*) and the exponential function (*pow*) are used. Instruction in lines 21 is intended to display the values of the horizontal distance with an accuracy of two decimal places.

In the line 23, the function *atan2l* is called, and the increments *ΔY, ΔX* are passed to this function as parameters. The return value of the function is recorded to an *angle* variable of the *long double* type. If as a result of arctangent calculation a negative number is obtained, i.e. an angle lying in the second or fourth mathematical quarter of the circle, then the value of the angle increases by 2π. To do this the constant M_PI which is equal to π is used.

Then the *convertingRadianToDegMinSec* function is called to convert the resulting radian value into an angle of degrees, minutes, and seconds. The function receives four parameters, and the angle expressed in radians, and three references on type *int* are among them. These three references will be directed to the variables of the same name in the body of the main function. As a result, the value of the bearing grid calculated in degrees, minutes and seconds will be recorded to the variables in the main function. Usage of references also solves the problem of returning multiple values from a function.

In order not to lose significant figures the variable *degreesWithFractPart* of the *long double* type are used. In the line 35 the integer part corresponding to the whole value of degrees is extracted from the value of the angle with the fractional part and the result is recorded to the *degrees* variable of the *main* function by reference. Similarly, the integer part of minute is extracted. The resulting value of minutes is recorded to the variable *minutes* of the main function by reference. Further, in the line 41 the extraction of an integer of seconds is similarly performed. The resulting value of seconds is recorded to the variable *seconds* of the main function by reference.

At the final stage of the function (lines 43–52), the discarded part of the angular seconds value is analyzed. If the discarded part is greater than 0.5" then the number of seconds obtained is increased by 1". If the total value of angular seconds has reached 60" then the number of minutes is increased by 1, and the second’s value is set to nil. If the total value of angular minutes has reached 60' as a result of increasing the number of angular minutes by 1, then the number of degrees is increased by 1, and the minute’s value is set to nil.

Suppose that the student has entered the following data into the program. The coordinates of point A are equal to $X_A = -256.23$ m and $Y_A = 300.18$ m, the coordinates of point B are equal to $X_B = 123.37$ m and $Y_B = -245.23$ m. As a result, the program will display the following on the screen:

```
Distance between two points:   664.51 m
Grid bearing of a line AB:     304° 50'15"
```

Running a program consisted of control data is an important condition when it comes to evaluation of student performance. Both skills of the C++ programming developed at the course and knowledge of general geodesy and cartography, are to be evaluated.

It should be noted about the relation between programming and other sciences included in geo-informatics. Junior students MIIGAiK learn programming in C, C++, Delphi and Python. The obtained knowledge and skills are the great help when it comes to work with GIS that perform geo-data processing. Obviously, one cannot do without having learned the basics of programming and computer science. In any programming language contain such control structures as conditional and unconditional jumps or loops. It is necessary to know and to use those constructions for writing macros for GIS. Good programming skills do the groundwork for studying geo-informatics in general and working with modern GIS.

4. Conclusions

The experience of teaching C++ programming language for students of cartographers and surveyors at the Moscow University of Geodesy and Cartography since 2009 to 2019 has been generalized. A new course of C++ programming for surveyors and cartographers has been developed. A wide range of training programs with the geodesy and cartography contents has been developed. The course is based on training geodetic tasks. On lectures and practical lessons an interactive electronic board is effectively used. Another technical feature aimed to increase the effectiveness of training is a remote administration program for computers linked up to a computer network. Using only MS-DOS environment while developing programs is dull for students and therefore this method of programming is not effective. So it is used only at the initial stage of training to illustrate the basic language scheme. A shift to Windows programming is made. An educational object-oriented program Theodolite is considered. The program has class "Theodolite" including the two constructors, destructor and method for visualization of technical parameters of theodolite. As example the training program for students studying the basics of C++ programming is discussed.

The experience of recent years shows that the level of school programming has increased significantly. If 15-20 years ago, first year MIIGAiK students had a very poor understanding of C++ programming, nowadays the completely opposite happens. The academic stuff has also undergone significant changes. So, if earlier a teacher of programming was experienced in computational mathematics and was little knowledgeable in applied disciplines, such as geodesy and cartography, now a teacher of programming is a young person, often, a graduate of the same university he works for. In this case, he is quite familiar with the basic subjects of this educational institution and can work with a wide range of specialized programs, as well as those which are related to cartography and geodesy. We assume that in the nearest future, the process of separation of programming according to branches of learning in the field of engineering. It is very likely that similar trends take place in other educational institutions not only in those related to geodesy and cartography.

5. References

Bronson, G. (2013). C++ for Engineers and Scientists. 4th Edition Published by Cengage Learning, 20 Channel Center Street, Boston, MA 02210, USA.

Gottschling, P. (2016). Discovering Modern C++. An Intensive Course for Scientists, Engineers and Programmers. Published by Addison–Wesley.

Liberty, J. and Cashman, M. (2002). SAMS Teach Yourself C++ in 10 Minutes, 2nd Edition, Published by SAMS, 800 East 96th Street, Indianapolis, Indiana, 46240 USA.

Liberty, J. and Cadenhead, R. (2011). SAMS Teach Yourself C++ in 24 Hours, 5th Edition, Published by SAMS, 800 East 96th Street, Indianapolis, Indiana, 46240 USA.

Prata, S. (2011). C++ Primer Plus, 6th Edition, Published by Addison–Wesley.

Schildt, H. (2003). C++ from the Ground Up, McGraw-Hill/Osborne, Berkley, CA.

Stroustrup, B. (1997). The C++ Programming Language. Special Edition, Addison-Wesley, Reading, MA.

Zablotskii,V. (2009). Learning the C/C++ language on the basis of programming geodetic tasks. A collection of articles prepared to The international scientific-technical conference, dedicated to the 230th anniversary of MIIGAiK, issue 2, part 1, Moscow: MIIGAiK, 199-202.