http://dx.doi.org/10.15576/GLL/2023.3.31

GLL

Geomatics, Landmanagement and Landscape No. 3 • 2023, 31–46

THE POINT LOCATION PROBLEM ON THE EXAMPLE OF DETERMINING MAP IDENTIFICATION NUMBER AND MAP SHEET EXTENT IN THE IMW SYSTEM

Mariusz Zygmunt, Krystyna Michałowska, Marek Ślusarski

Summary

The point location problem is one of the most fundamental topics in computational geometry. This problem is particularly well illustrated in the context of maps. In order to determine a point location, it is necessary to work in a specified space, with known division rules. A practical example can be a map sheet division system adopted for the International Map of the World. Given the current advancements in spatial data acquisition and growing availability of spatial data sets, it has become necessary to design an algorithm to identify a map sheet corresponding to a selected point. Available digital data, e.g. orthophotomaps or LiDAR-based data sets, cover whole countries. Due to the size of such data sets, they need to be divided into smaller chunks. The article presents a method of determining a map sheet identification number and a map sheet extent based on the latitude and longitude of a selected point. An inverse problem has also been addressed, allowing a map sheet extent to be identified based on a map sheet identification number. The algorithms developed might be directly implemented in GIS software. They are presented in a ready-to-implement form in the Basic programming language with the use of basic data structures.

Keywords

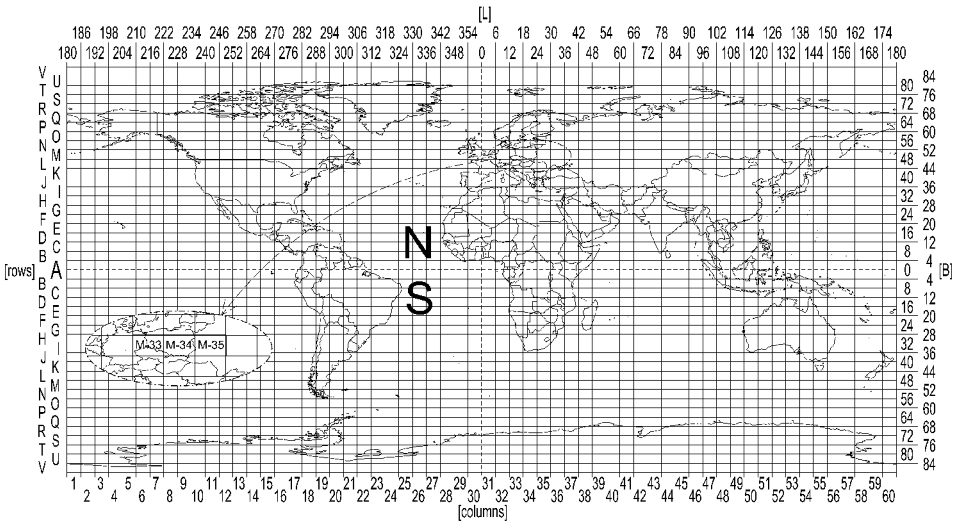
IMW • index map • geometrical geodesy

1. Introduction

The point location problem has been tackled over the years by numerous researchers [Preparata and Shamos 1985, Snoeyink 2004]. It can be classified into four main approaches [De Berg et al. 2008]:

a) storage solutions [Edelsbrunner et al. 1986],
b) triangulation refinement method [Kirkpatrick 1983],
c) use of persistency [Sarnak and Tarjan 1986, Cole 1986],
d) randomised incremental method [Mulmuley 1990].

This article focuses on the planar point location problem. The issue is discussed in context of maps divided into sheets in accordance with the International Map of the World (IMW). In this system, the planar point location problem can be described as finding point locations on a plane divided into rectangular grid cells [Bentley 1975, Lueker 1978, Willard and Lueker 1985, Karlsson and Munro 1985, Karlsson and Overmars 1988, Samet 1990, Overmars 1988, Samet 1990]. The concept of the division is based on the idea introduced by Albrecht Penck (1858–1945), who proposed to divide the Earth into 2,500 map sheets at a 1:1,000,000 scale (Fig. 1). Each grid cell was six degrees longitude by four degrees latitude [Maps. Boston Public Library]. Although the project was not been fully completed, the assumptions of this map sheet division are used worldwide [Parry and Perkins 2011]. Its modified version is currently in use in the Global Mapping Project (GMP) [Pearson et al. 2006], and other up-to-date cartographic representations are also based on the IMW grid, e.g. the World Aeronautical Chart (WAC).



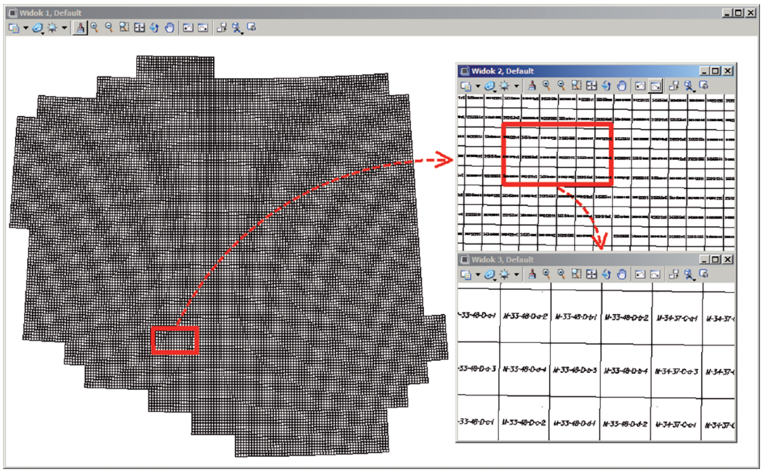
Source: Authors' own study

Fig. 1. Division of the world map into sheets at a 1:1,000,000 scale

Spatial data sets can be organised and managed in various ways, but efficiency is a key factor. Considering different types of data storage and management, the following types of databases can be distinguished [Longley et al. 2005]:

a) relational (Relational Database Management System – RDBMS),
b) object (Object Database Management System – ODBMS),
c) object-relational (Object-Relational Database Management System – ORDBMS).

Object-relational databases store data on spatial objects in two-dimensional tables. In order to find a map sheet to which a given point belongs, in a typical database it is necessary to provide coordinates of boundaries of all map sheets within a search extent. Such an approach is space-consuming and in the case of dividing map sheets into larger scales, it requires storing further information on their extents. Figure 2 presents how much data needs to be stored in the case of Poland divided into map sheets at a 1:10,000 scale (18,950 rectangular cells). A subsequent division into a 1:5,000 scale quadruples the number of map sheets, and for a 1:2,500 scale the previous number of map sheets is quadrupled again, reaching the number of 303,200 rectangular cells in total. Such a scale is used in the division of LiDAR-based data which covers the whole territory of Poland (312,679 km^2), which enables its more efficient usage. Related issues are also important in view of navigation, when a map sheet corresponding to a point location needs to be downloaded.



Source: Authors' own study

Fig. 2. Volume of data for the division of Poland into 1:10,000 map sheets – traditional data processing approach would require constructing tables with indexes for all map sheets

In order to ensure efficient data processing and access, the mechanism of spatial indexing needs to be implemented. The most commonly used approaches include [Longley et al. 2005]:

a) Balanced trees,
b) Grid indexes,
c) Quadtrees,
d) R-trees.

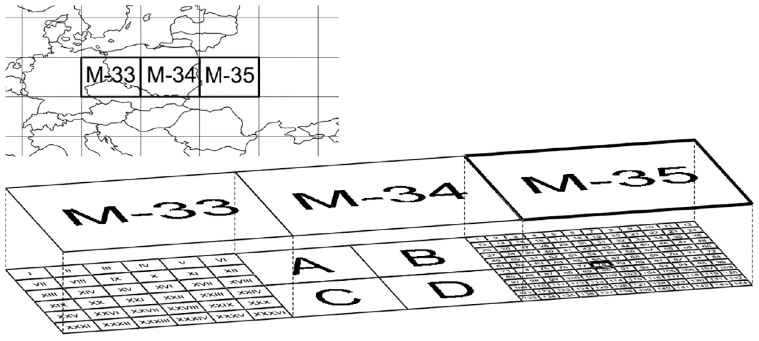
Unfortunately, all these approaches require storing information on spatial indexes. One of the main goals of this research was to provide a more efficient solution for any amount of data. The solution presented in this paper belongs to the field of computational geometry, as it concerns research on a novel, algorithmic approach towards the automation of the process of planar point location and map sheet identification [Preparata and Shamos 1985, Snoeyink 2004]. The issue of point location in more than two dimensions was not discussed, as a search extent was limited to planar dimensions.

2. Materials and methods

2.1. Map sheet division at various scales

A basic map sheet at a scale of 1:1,000,000 covers an area of six degrees longitude by four degrees latitude. Division in a longitudinal direction forms columns described with numbers from 1 to 60. The point of origin is the 180th meridian (antimeridian). Division in a latitudinal direction creates rows described with letters from A to Z, starting from the Equator and continuing in both the north and the south direction, independently. The hemispheres can be distinguished based on a letter which appears in a map sheet identification number – N (Northern Hemisphere) or S (Southern Hemisphere). The examples of map identification numbers at a scale of 1:1,000,000 can be N-M-35 for the Northern Hemisphere and S-M-35 for the Southern Hemisphere. In the case of maps presenting areas located on only one of the hemispheres, the identifying letter is frequently omitted. This type of identification mostly applies at country levels.

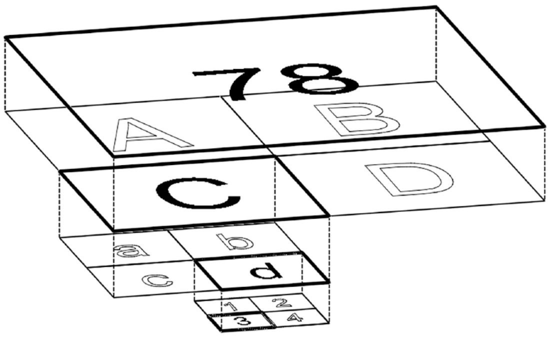
Map sheet division at a scale of 1:1,000,000 is carried out based on a scheme presented in Figure 3. For the scale of 1:500,000, it is necessary to divide a 1:1,000,000 sheet into four parts, assigned with letters A, B, C and D, e.g. N-M-34-A. A map sheet identification number at a scale of 1:200,000 is given based on the division of a 1:1,000,000 map sheet into 36 sheets marked with subsequent Roman numerals, e.g. N-M-33-XVI. In order to obtain a map sheet identification number at a scale of 1:100,000, a 1:1,000,000 map sheet needs to be divided into 144 sheets described with subsequent Arabic numerals, e.g. N-M-35-78.



Source: Authors' own study

Fig. 3. Division of a basic map sheet at a scale of 1:1,000,000. Left division for the scale of 1:200,000 (I–XXXVI), middle division for the scale of 1:500,000 (A–D), right division for the scale of 1:100,000 (1–144) – the sheet selected is no. 78

Division for larger scales is conducted based on a 1:100,000 sheet. Map identification numbers for the scale of 1:50,000 are obtained by dividing a 1:100,000 sheet into 4 parts marked with letters A, B, C and D, e.g. N-M-35-78-C. A further division of a 1:50,000 map sheet into 4 parts, assigned with letters a, b, c and d, results in obtaining a map sheet identification number at a scale of 1:25,000, e.g. N-M-35-78-C-d. Map identification numbers at a scale of 1:10,000 are obtained by dividing a 1:25,000 sheet into 4 parts marked with numbers 1, 2, 3 and 4, e.g. N-M-35-78-C-d-3 (Fig. 4).

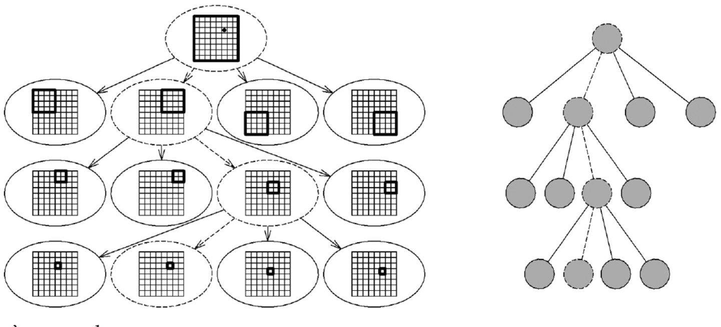


Source: Authors' own study

Fig. 4. Division of a map sheet no. 78 at a scale of 1:100,000

2.2. The main concept of the algirthms

The main concept of the algorithms is based on creating spatial indexes by using quadtrees. Quadtrees are data structures proposed in 1974 by informatician, Raphael Finkel, and mathematician, John Louis Bentley [Finkel and Bentley 1974]. This structure is constructed by subsequent, recursive divisions of space into square fields. Two—dimensional space is divided into 4 parts, which are divided into another 4 parts, etc. (Fig. 5).



Source: Authors' own study

Fig. 5. Division of two-dimensional space with a quadtree model; dashed-line boxes and dashed arrows indicate subsequent divisions needed for determining the location of a selected point (marked as a black dot in a top box)

The first developed algorithm was inspired by the concept of recursive data processing within the map sheet extent with determined BL boundary coordinates (Table 1). In this case, the concept of dividing space into 4 parts was further developed. The number of divisions is defined by the map scale, for which a map sheet identification number is to be determined. In the first step, the algorithm determines the location of a selected point after dividing the map into 2640 map sheets (60 × 22 × 2) (Fig. 1). The division is performed based on a rectangular grid, whose dimensions are determined by the BL coordinates of map sheets. The next steps depend on the final division which is to be achieved, i.e. the map sheet identification number to be obtained, which is related to the desired map scale. The algorithm calculates iteratively the coordinates of the map sheet corners and the alphanumeric characters of a map sheet identification number for a given scale. Next, it examines the location of a searched point within a smaller map sheet, based on the adapted division model, e.g. into 144, 36, or 4 sheets. Subsequent divisions are restricted by the final scale, and after each iteration a new character is added to the map sheet identification number. For the scale of 1:100,000, two divisions and updates of the map sheet identifaction number are performed, for the scale of 1:2,500 there are 7 such operations. Map sheets of scales larger than 1:100,000 are always divided into 4 parts. This is why the developed algorithm has no limitations in determining the map sheet identification numbers in larger scales, which are not described in the literature. This issue results, among others, from the need for assigning large data sets (e.g. LiDAR data) to the areas with set boundary coordinates. It is assumed that the subsequent divisions are performed according to the same order.

The inverse algorithm (Table 2) allows a map sheet extent to be calculated using a map sheet identification number, with the same iterative method. After determining a map sheet extent in the given scale, the increments of coordinates for subsequent scales are calculated. The computation starts with determining the coordinates of a map sheet corner at the scale of 1:1,000,000. Then the local coordinates of other map sheet corners at different scales within a map sheet extent at the scale of 1:1,000,000 are determined. The final step is the reduction of the map sheet coordinates with accordance to the calculated increments in order to obtain real BL coordinates in the IMW system.

2.3. An algorithm for calculating map sheet identification numbers and map sheet extents

The algorithm for calculating map sheet identification numbers is described in detail in [Zygmunt et al. 2021]. However, given various practical conditions, it turned out that calculating a map sheet identification number alone is not sufficient. Since it is necessary to visualise a map sheet extent in a specified cartographic projection, it is necessary to provide its boundaries in a given coordinate reference system. The algorithm presented in [Zygmunt et al. 2021] has been refined and further developed. The newly created part of the code is typed with a bold font in lines 4 and 5, which call the function, and lines 55-70 that define calculation methods (Table 1).

Table 1. The algorithm for calculating map identification numbers

```
Function Get_INDEX_IMW_Frame _
    (L As Double, B As Double, _
    ScaleID As Integer, N_S As Integer, _
    L_min As Double, B_min As Double, _
    L_max As Double, B_max As Double) _
    As String
  Dim INDEX As String
  Dim dL As Double, dB As Double
  Dim L0 As Double, B0 As Double
  Dim Row As Integer, Column As Integer
  Dim i As Integer
  If L < 0 Or L > 360 Or B > 88 Then
  Get_INDEX_IMW_Frame = "Bad data": Exit Function
  End If
  If N_S = 1 Then INDEX = "N-" Else INDEX = "S-"
  For i = 0 To ScaleID
    If ScaleID = 2 And i = 1 Then i = 2
    If ScaleID > 2 And i = 1 Then i = 3
    If i < 4 Then
      dL = 6
    Else
      If i = 4 Then dL = 1
      dL = dL / 2
    End If
    dB = dL * 2 / 3
    L0 = Int(L / dL) * dL
    B0 = Int(B / dB) * dB
    If N_S = 1 Then B0 = B0 + dB
    If i = 0 Then
      Row = Int(B / dB)
      Column = Int((L + 186) / dL)
      If Column > 60 Then Column = Column – 60
      INDEX = INDEX & Chr(65 + Row) & "-" & CStr(Column)
  ElseIf i = 2 Then
    Row = Int(N_S * (B0 – B) / (2 / 3))
    Column = Int(L – L0)
    INDEX = INDEX & "-" & Get_Roman_1_36(Row * 6 + Column + 1)
    ElseIf i = 3 Then
    Row = Int(N_S * (B0 – B) * 3)
    Column = Int((L – L0) * 2)
    INDEX = INDEX & "-" & CStr(Row * 12 + Column + 1)
    Else
      Dim Start As Integer
      Row = Int(N_S * (B0 – B) / (dB / 2))
      Column = Int((L – L0) / (dL / 2))
```

Table 1. cont.

```
46.       If i = 1 Or i = 4 Then
47.       Start = 65
48.     ElseIf i = 5 Then
49.       Start = 97
50.     Else
51.       Start = 49
52.     End If
53.     INDEX = INDEX & "-" & Chr(Start + Row * 2 + Column)
54. End If
55.       If i = 0 Then
56.       L_min = L0
57.       B_min = N_S * B0 – dB
58.       L_max = L_min + dL
59.       B_max = B_min + dB
60.   Else
61.       Dim n As Integer
62.       n = 6 * (i – 1)
63.       If i = 1 Or i > 3 Then n = 2
64.       L_min = L0 + Column * dL / n
65.       B_min = N_S * B0 – (Row + 1) * dB / n
66.       L_max = L_min + dL / n
67.       B_max = B_min + dB / n
68.   End If
69.   B_min = Abs(B_min)
70.   B_max = Abs(B_max)
71.   Next i
72.       Get_INDEX_IMW_Frame = INDEX
73. End Function
```

Source: Authors' own study

The newly created function Get_INDEX_IMW_Frame returns a map sheet identification number and, additionally, L_min, B_min, L_max and B_max coordinates describing the minimum and maximum longitude and latitude of a map sheet extent. In lines 55–70, the algorithm calculates iteratively map sheet extents corresponding to consecutive map sheet divisions at different scales.

2.4. An algorithm for calculating map sheet extents based on map identification numbers

The new algorithm for calculating map sheet extents based on map identification numbers is presented in Table 2.

Table 2. Algorithm for calculating map sheet extents based on map identification numbers

```
Sub Put_Frame_IMW _
    (N_S As String, Row As String, Column As Integer, _
    Data_144_36 As Integer, DataU_ABCD As String, _
    DataL_abcd As String, Data_1234 As Integer, _
    ScaleID As Integer, _
    L_min As Double, B_min As Double, _
    L_max As Double, B_max As Double) As String
  Dim L As Double, B As Double
  Dim dL As Double, dB As Double
  Dim deltaL As Double, deltaB As Double
  Dim Nr As Integer, i As Integer, Lp As Integer
  B = (Asc(Row) – 64) * 4
  L = Column * 6 – 186
  If L < 0 Then L = L + 360
  dL = 6: dB = 4: Lp = 1
  For i = 1 To ScaleID
    If ScaleID = 2 Then i = 2 ‘200 000
    If ScaleID > 2 And i < 3 Then i = 3 ‘100 000
    If i = 1 Or i = 4 Then ‘500 000, 50 000
      Nr = Asc(DataU_ABCD) - 64 ‘A,B,C,D
      Lp = 2
      If i = 4 Then dL = 1
    ElseIf i = 2 Or i = 3 Then ‘2=I-XXXVI(1–36), 3=1–144
      Nr = Data_144_36
      Lp = (i – 1) * 6
    ElseIf i = 5 Then ‘1:25 000
      Nr = Asc(DataL_abcd) – 96 ‘a,b,c,d
    Else ‘1:10 000
      Nr = Data_1234 ‘1,2,3,4
    End If
    If i > 3 Then dL = dL / 2
    dB = dL * 2 / 3
    deltaL = deltaL + (Nr – (Int((Nr – 1) / Lp)) * Lp – 1) * (dL / Lp)
    deltaB = deltaB + Int((Nr – 1) / Lp) * (dB / Lp)
  Next i
  L_min = L + deltaL
  B_min = B – deltaB – dB / Lp
  L_max = L_min + dL / Lp
  B_max = B_min + dB / Lp
  If N_S = “S” Then
    B_max = B_max – (2 * B – 4)
    B_min = B_max – dB / Lp
  End If
End Sub
```

Source: Authors’ own study

Line 1 contains the name of the procedure. Lines 2–5 define the input parameters which are required for the algorithm to work. Lines 6 and 7 define the output variables. Local variables are declared in lines 8–11, and lines 12–15 are responsible for computations and initial variable substitutions. The most important computational part of the algorithm is included in lines 16–35, followed by lines 36–43, which prepare the final outputs.

Line 2. The **N_S** variable requires the input of two values – N for the Northern Hemisphere or S for the Southern Hemisphere. **Row** indicates a corresponding row in the IMW system described by an upper case letter. **Column** is a value which indicates a corresponding column, marked with a number from 1 to 60.

Line 3. The **Data_144_36** variable indicates the number (1–144) of a basic map sheet section at a scale of 1:100,000, or the number of a map sheet section at a scale of 1:200,000. As adopted in the IMW system, section numbers at a 1:200,000 scale should be given in the Roman system. The presented algorithm requires conversion of such numerals to the Arabic system. **DataU_ABCD** provides a map sheet identification number at a scale of 1:500,000 or 1:50,000. The input values must be the letters A, B, C or D, typed in uppercase.

Line 4. The **DataL_abcd** variable indicates the number of a map sheet at a scale of 1:25,000. The input values must be the letters a, b, c or d, typed in lowercase. The **Data_1234** variable is used for the number of a map sheet at a scale of 1:10,000. The values must be the numerals 1, 2, 3, or 4.

Line 5. The **ScaleID** variable defines the scale for which boundary coordinates of a map sheet should be prepared. Acceptable values are the integer numbers from 0 to 6, which apply for the following scales: 0 – 1:1,000,000, 1 – 1:500,000, 2 – 1:200,000, 3 – 1:100,000, 4 – 1:50,000, 5 – 1:25,000 and 6 – 1:10,000.

Lines 6, 7. These lines define the output variables. **L_L** defines a longitudinal coordinate of a left boundary of a map sheet, **B_B** defines a latitudinal coordinate of a bottom boundary, **L_R** defines a longitudinal coordinate of a right boundary, and **B_T** defines a latitudinal coordinate of a top boundary.

Line 8. Declaration of variables: **L** – longitude, and **B** – latitude of a basic map sheet at a scale of 1:1,000,000.

Line 9. Declaration of variables: **dL** – map sheet extent in relation to longitude, and **dB** – in relation to latitude.

Line 10. Declaration of variables: **deltaL** – defines the shift of a map sheet origin in relation to longitude, **deltaB** – defines the shift of a map sheet origin in relation to latitude.

Line 11. Declaration of variables: **Nr** – number of a consecutive map sheet, **i** – main loop counter, **Lp** – number of divisions of a basic map sheet into larger-scale sheets.

Line 12. Calculation of latitude for a basic map sheet based on the value given in the **Row** field. The **Asc** function returns the number assigned to a typed letter. The obtained value minus 64 organises the values from 1 in an ascending order. When multiplied by 4, it gives the final output.

Line 13. Calculation of longitude for a basic map sheet. Given the longitude of the anti-meridian, it is necessary to reduce the calculated longitude by 186 degrees.

Line 14. If the value obtained in **Line 13** is a negative number, this line is used to transform it into a positive one.

Line 15. Declaration of values for a basic map sheet frame extent at 1:1,000,000 scale.

Line 16. Start of the main loop of the algorithm.

Line 17. If the **ScaleID** variable equals 2, the loop counter **i** is set to 2. This way, computations are performed immediately for the scale of 1:200,000, and the division of a map sheet into 36 parts is conducted.

Line 18. If the **ScaleID** variable is higher than 2, the loop counter **i** is set to 3. This operation allows for omitting unnecessary division of a basic map sheet into 4 parts at 1:500,000 scale, or 36 parts at 1:200,000 scale.

Line 19. Start of a conditional statement for the divisions into 1:500,000 and 1:50,000 scale.

Line 20. Calculation of a map sheet number for a selected scale: A = 1, B = 2, C = 3, D = 4.

Line 21. Setting the number for a map sheet division. For the selected scales the value is 2.

Line 22. Setting the length of a sheet to 1 for 1:50,000 scale.

Line 23. Start of a conditional statement for the division into 1:200,000 and 1:100,000 scale.

Line 24. Assigning the **Nr** variable with a map sheet number from 1 to 144.

Line 25. Calculation of a number of divisions for the scale of 1:100,000. For **i** = 2 (1:200,000 scale), the **Lp** variable equals 6, and for **i** = 3 (1:100,000 scale), **Lp** = 12.

Line 26. Start of a conditional statement for the division into 1:25,000 scale.

Line 27. Calculation of a map sheet number for a selected scale: a = 1, b = 2, c = 3, d = 4.

Line 28. Start of a conditional statement for the division into 1:10,000 scale.

Line 29. Assigning the **Nr** variable with a map sheet number from 1 to 4.

Line 30. End of a conditional statement.

Line 31. Conditional statement establishing subsequent divisions of a map sheet extent into 2 parts at scales: 1:50,000, 1:25,000 and 1:10,000.

Line 32. Calculation of a map sheet extent in relation to latitude. The 2/3 ratio results from the proportions of a basic sheet dimensions.

Line 33. Calculation of a total increment in a longitudinal value of a map sheet origin, calculated in a local coordinate system, as a sum of subtotal longitudinal map sheet values in particular scales.

Line 34. Calculation of a total increment in a latitudinal value of a map sheet origin, calculated in a local coordinate system, as a sum of subtotal latitudinal map sheet values in particular scales.

Line 35. End of the main loop of the algorithm.

Line 36. Calculation of a longitudinal coordinate for a left map sheet boundary.

Line 37. Calculation of a latitudinal coordinate for a bottom map sheet boundary.

Line 38. Calculation of a longitudinal coordinate for a right map sheet boundary.

Line 39. Calculation of a latitudinal coordinate for a top map sheet boundary.

Line 40. Conditional statement executed in the case of calculations for the Southern Hemisphere.

Line 41. Reduction of latitude of a top map sheet boundary.

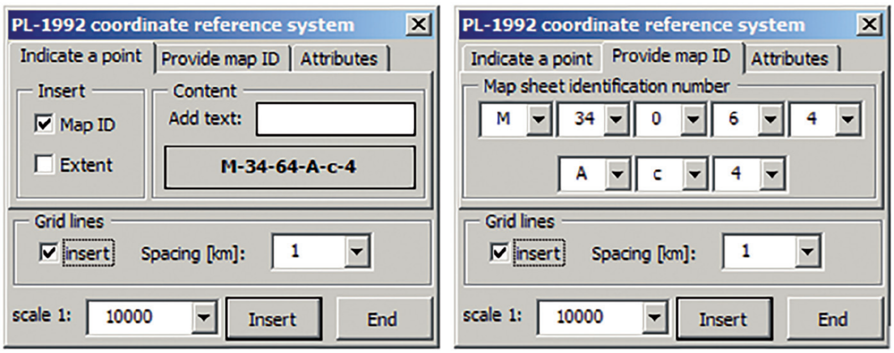
Line 42. Reduction of latitude of a bottom map sheet boundary.

Line 43. End of the conditional statement.

Line 44. End of the procedure of calculating a map sheet extent.

3. Results

The developed algorithm was implemented in CAD/GIS software to facilitate works with data sets in the PL-1992 coordinate reference system (EPSG:2180) (Fig. 6). This reference system uses planar XY coordinates. However, the division into map sheet sections, and their identification are based on the IMW system and BL coordinates. The interface of the program was designed in two versions. The first one is dedicated to calculations based on BL coordinates (requires a point to be indicated with the mouse) and the other uses a map identification number in the IMW system. Both versions can be complemented with a kilometre grid generated for an analysed area.


PL-1992 coordinate reference system

Insert: ☑ Map ID ☐ Extent

Content: Add text:

M-34-64-A-c-4

Grid lines: ☑ insert Spacing [km]: 1

scale 1: 10000

PL-1992 coordinate reference system

Map sheet identification number: M 34 0 6 4 A c 4

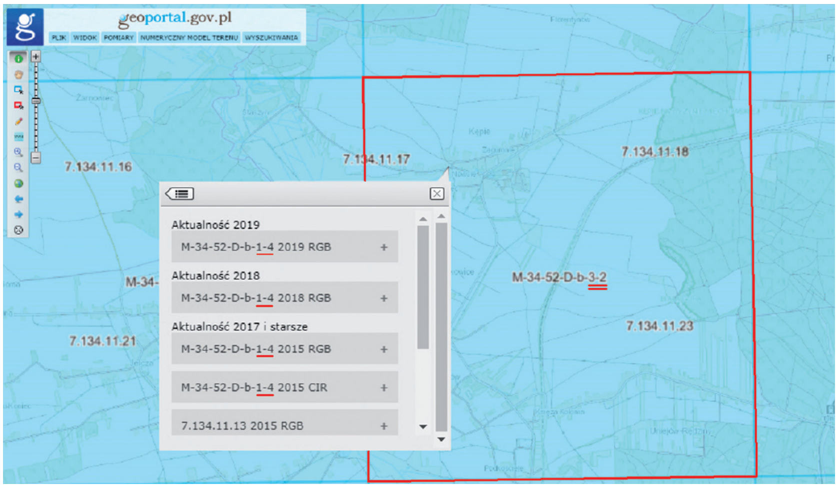
Grid lines: ☑ insert Spacing [km]: 1

scale 1: 10000

Source: Authors' own study

Fig. 6. A practical implementation of the algorithm: the interface of the program facilitating works with map sheets in CAD software

During the works concerning data acquisition from 'Geoportal, https://mapy.geoportal.gov.pl' – a national database – several errors resulting from misinterpretation of data sets identified by a map sheet identification number in the IMW system have been detected. For the map section M-34-54-D-b-3-2, presented in Figure 7, the wrong data sets have been uploaded (from the M-34-52-D-b-1-4 section). The algorithms described in this paper may help to verify existing data sets.



Aktualność 2019
M-34-52-D-b-1-4 2019 RGB +
Aktualność 2018
M-34-52-D-b-1-4 2018 RGB +
Aktualność 2017 i starsze
M-34-52-D-b-1-4 2015 RGB +
M-34-52-D-b-1-4 2015 CIR +
7.134.11.13 2015 RGB +

Source: Authors' own study

Fig. 7. An error resulting from misassignment of data to data sets with a specified map identification number

4. Discussion

Map sheet division in the IMW system presented in this paper is not the only one used in the world. Algorithms need to be modified and adapted to specific conditions and selected scales. The procedure described in the article is not suitable for calculations in the areas with latitudes greater than 60 degrees. Due to reductions related to cartographic projections, the horizontal extent of a map sheet is doubled. For latitudes greater than 76 degrees, the sheet extent increases by 24 degrees. In such cases, adjacent map sheets are joined together.

The Get_Roman_1_36 function (Table 3), which ensures correct calculations at a 1:200,000 scale can be replaced by a 36-element table filled with string elements. Subsequent elements of the table are assigned their Arabic equivalents. This paper uses a function originally developed by the author. The function has been tested for a limited range of numbers, from 1 to 36.

Table 3. The Get_Roman_1_36 function

```
Function Get_Roman_1_36(arabic As Integer) As String
  Dim LArray(9) As String
  Dim roman As String
  Dim n As Integer

  LArray(0) = “”
  LArray(1) = “I”
  LArray(2) = “II”
  LArray(3) = “III”
  LArray(4) = “IV”
  LArray(5) = “V”
  LArray(6) = “VI”
  LArray(7) = “VII”
  LArray(8) = “VIII”
  LArray(9) = “IX”
    If arabic > 29 Then
    roman = “XXX”
  ElseIf arabic > 19 Then
    roman = “XX”
  ElseIf arabic > 9 Then
    roman = “X”
  Else
    roman = “”
  End If
    n = arabic – (10 * Len(roman))
    Get_Roman_1_36 = roman & LArray(n)
End Function
```

Source: Authors’ own study

5. Conclusions

The article presents a method for calculating map sheet identification numbers and map sheet extents for a selected point in the IMW system. The inverse problem was also addressed, allowing a map sheet extent to be determined based on BL coordinates, using a map sheet identification number. Such works fall within the scope of computational geometry. The developed algorithms have practical application potential, as they can be used in works concerning spatial data sets, e.g. raster data sets or point clouds. In real conditions, such data sets are often described with a map sheet number in the IMW system. The use of the developed algorithms significantly facilitates management of such data sets, especially in the case of large-area data sets. Thanks to the presented methodology, it is not necessary to create tables with spatial indexes in relational databases, which significantly reduces data volume. Another obstacle that arises while working with large data sets, e.g. LiDAR data, which was addressed in the paper, is the need for consecutive divisions of map sheets into smaller chunks. The solution proposed for determining a map sheet identification number and a map sheet extent ensures that the divisions are maintained at any scale.

Funded from a subsidy by the Ministry of Education and Science for the University of Agriculture in Krakow, for the year 2023.

References

Bentley J.L. 1975. Multidimensional binary search trees used for associative searching. Commun. ACM, 18(9), 509–517.

Cole R. 1986. Searching and storing similar lists. J. Algorithms, 7(2), 202–220.

De Berg M., Cheong O., van Kreveld M., Overmars M. 2008. Computational Geometry Algorithms and Applications, 3rd ed. Springer-Verlag Telos, Santa Clara, CA, USA.

Edelsbrunner H., Guibas L.J., Stolfi J. 1986. Optimal Point Location in a Monotone Subdivision. SIAM J. Comput., 15(2), 317–340.

Finkel R.A., Bentley J.L. 1974. Quad trees a data structure for retrieval on composite keys. Acta Informatica, 4, Springer, Berlin.

Karlsson R.G., Munro J.I. 1985. Proximity on a grid. Proceedings on STACS 85 2nd Annual Symposium on Theoretical Aspects of Computer Science. Lecture Notes in Computer Science, Springer-Verlag, 182, 187–196.

Karlsson R.G., Overmars M.H. 1988. Scanline algorithms on a grid. BIT Numeral Mathematics, 28, 227–241.

Kirkpatrick D. 1983. Optimal Search in Planar Subdivisions. SIAM J. Comput., 12(1), 28–35. https://doi.org/10.1137/0215023.

Longley P.A., Goodchild M.F., Maguire D.J., Rhind D.W. 2005. Geographic Information Systems and Science. John Wiley & Sons.

Lueker G.S. 1978. A data structure for orthogonal range queries. Proceedings on 19th Annu. IEEE Sympos. Found. Comput. Sci., 28–34.

Maps. Boston Public Library. http://maps.bpl.org/id/m8799.

Mulmuley K. 1990. A fast planar partition algorithm, I. Journal of Symbolic Computation, 10(3–4), 253–280. https://doi.org/10.1016/S0747-7171(08)80064-8.

Overmars M.H. 1988. Efficient data structures for range searching on a grid. J. Algorithms, 9(2), 254–275.

Parry R.B., Perkins C.R. 2011. World mapping today, 2nd ed. De Gruyter.

Pearson A., Taylor D.R.F., Kline K.D., Heffernan M. 2006. Cartographic ideals and geopolitical realities: International maps of the world from the 1890s to the present. The Canadian Geographer, 50(2), 149–176. https://doi.org/10.1111/j.0008-3658.2006.00133.x

Preparata F.P., Shamos M.I. 1985. Computational Geometry: An Introduction. Springer-Verlag.

Samet H. 1990. Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, US.

Samet H. 1990. The Design and Analysis of Spatial Data Structures. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, US.

Sarnak N., Tarjan R.E. 1986. Planar point location using persistent search trees. Commun. ACM, 29(7), 669–679. https://doi.org/10.1145/6138.6151.

Snoeyink J. 2004. Point location. In: Handbook of Discrete and Computational Geometry, 2nd ed. Eds. J.E. Goodman, J. O'Rourke. CRC Press LLC, Boca Raton, FL, USA.

Willard D.E., Lueker G.S. 1985. Adding range restriction capability to dynamic data structures. J. ACM, 32(3), 597–617.

Zygmunt M., Gniadek J., Szewczyk R. 2021. The method for setting map sheet identification numbers in the International Map of the World (IMW) system. Geomatics, Landmanagement and Landscape, 1, 69–79. https://doi.org/10.15576/GLL/2021.1.69

Dr hab. inż. Mariusz Zygmunt
University of Agriculture in Krakow
Department of Goedesy
30-198 Kraków, ul. Balicka 253a
e-mail: mariusz.zygmunt@urk.edu.pl
ORCID: 0000-0003-0056-5215

Dr inż. Krystyna Michałowska
University of Agriculture in Krakow
Department of Goedesy
30-198 Kraków, ul. Balicka 253a
e-mail: krystyna.michalowska@urk.edu.pl
ORCID: 0000-0001-7749-3622

Dr hab. inż. Marek Ślusarski
University of Agriculture in Krakow
Department of Goedesy
30-198 Kraków, ul. Balicka 253a
e-mail: marek.slusarski@urk.edu.pl
ORCID: 0000-0002-8573-936X