INTRODUCTION

One of the main tasks of information systems is the search of needed information. It is important to be economical about the data presentation and, moreover, be effective with information search without wasting time and effort. In order to solve this key problem a number of different approaches were formed, one of them is search trees.

There are different types of search trees, and one of the major ones as well as the one our algorithm was based on is binary trees.

Binary tree is the tree with no more than two edges going from each vertex, including the root. In this case, each vertex(except for root) of the binary tree as an input edge has no more than one edge.

Binary search trees is a binary tree for which following properties should be applied:

- Both subtrees (left and right) must be binary search trees
- The left subtree of a node reserves node with key values less than the node's key
- The right subtree of a node reserves node with key values more than the node's key

The main advantage of binary search tree over other data structures is the possible realization of high efficiency based on its search and sorting algorithms. However, binary tree has a number of disadvantages. One of them is that binary trees can only search information recorded in the form of one-dimensional array. But tree is a two-dimensional structure, therefore, it would be much more desirable for binary trees to handle two, three and even k-dimensional objects. This is one of our goals we will try to achieve.

The main difficulty of our thesis is the grid construction. The aim of our work is to localize the point, in other words to find the smallest rectangle this point is set. In order to avoid construction of grid with a lot of unnecessary lines, it would be better to construct grids recursively; i.e. after the first call find the minimum square which the point is set to. Then repeat construction of grid and determine the smallest square again. Procedure must be repeated until the acceptable point of localization is determined.

This project work can be helpful for different services like ambulance, police and fire protection. The same result can be obtained by brute force, but it will require much more time. For a small number of queries, the difference may not seem significant, however, if the number of requests will reach thousand or more, the difference becomes significant.

1 ANALYSIS OF THE EXISTING SEARCH TREES

1.1 Binary tree

The binary tree is a tree from each of the vertices of which, including the root, goes not more than two edges. In this case, each vertex(but root) of the binary tree as an input edge has, obviously, not more than one edge.

We have to emphasize that in many applications while construction of the tree it is very important to keep levels, i.e. vertices of the one level must be on one line, not necessarily, but most of all, on the horizontal.

Compliance of the levels helps in so-called traversing of the tree, one of the standard and important operations carried out over the trees. Note that the traversal (bypass) of the tree means consequent visit of all nodes (vertices) of the tree, noting each of them only once. There are three basic types of traversal – in order, symmetric and post order one.

1.1.1 Binary tree's types of traversal

The ordering of the array as a binary tree is not linear, i.e. it is not ordered in increasing or decreasing order. However, the question arises, what is the traversal of the tree nodes where the corresponding elements appear, for example, is in increasing order. This type of traversal is already mentioned the traversal of the tree in a symmetrical manner:

1) Bypass the left subtree follows in a symmetrical manner
2) Mark root (subtree, tree)
3) Bypass the right subtree follows in a symmetrical manner with supplemental recursion.

If we follow the rules 1-3 at traversal of the tree taking into account recursion, we exactly obtain an ordered array as an increasing sequence.

An example is shown in Figure1.1:

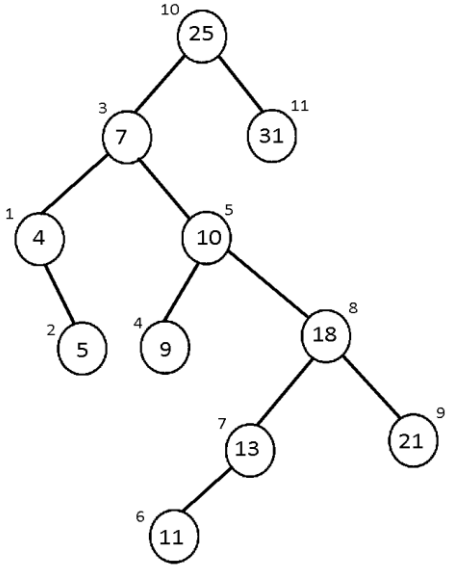

Figure 1.1 Preorder traversal

Result of example: *4,5,7,9,10,11,13,18,21,25,31*

Another type of bypass may seem more natural, without running far ahead, and therefore is called a preorder traversal, which is also defined recursively:

1) Mark root (subtree, tree)
2) Mark, not mark before the nodes of the left subtree follow in preorder traversal
3) Mark, not mark before the nodes of the right subtree follow in preorder traversal

An example is shown in Figure1.2,

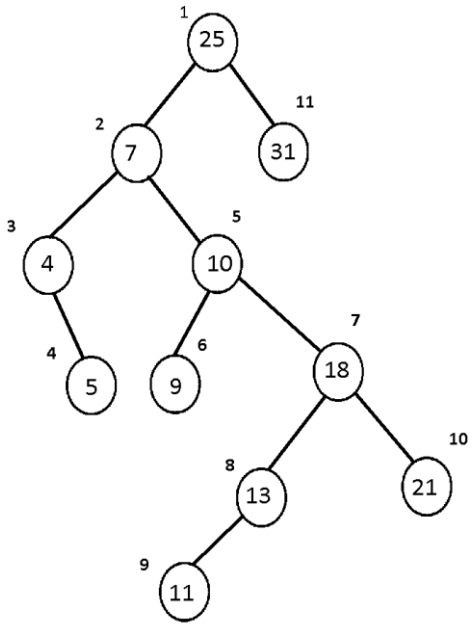

Figure 1.2 Postorder traversal

Result of example: *25,7,4,5,10,9,18,13,11,21,31*

The remaining type of traversal is a tree traversal in postorder manner:

1) Bypass in postorder manner left subtree
2) Bypass in postorder manner right subtree
3) Go to and mark the root (subtree, tree)

An example is shown in Figure1.3:

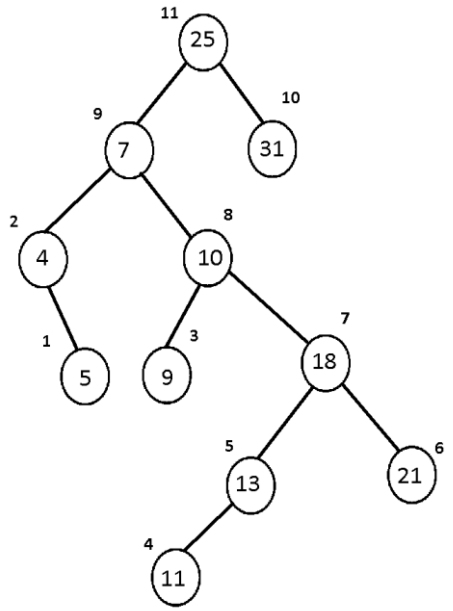

Figure 1.3 Symmetric traversal

Result of example: *5,4,9,11,13,21,18,10,7,31,25*

Each of the above types of traversal is useful for solving a particular problem associated with the use of binary trees. The usefulness of traversal in a symmetrical manner we've seen.

1.1.2 Special form of writing arithmetic expressions

Special form of writing arithmetic expressions initially containing operands, the signs of binary operations and, of course, the brackets. It is talking about the prefix, postfix and infix forms of recording an arithmetic expression. Each such expression can be associated with a binary tree. The root of the tree (subtree) contains the sign operation which is applied to the left and the right of direct descendants of the roots, which by nature are the operands of the operation. We must stress that it is possible to write these expression in the bracket-free form. In this case, we should not create the impression that, to determine and record the arithmetic expressions in prefix, postfix and infix forms we must first build a binary tree of these expressions. Although, if any, the task of constructing these forms are becoming more evident under visualization.

An example is shown in Figure1.4:

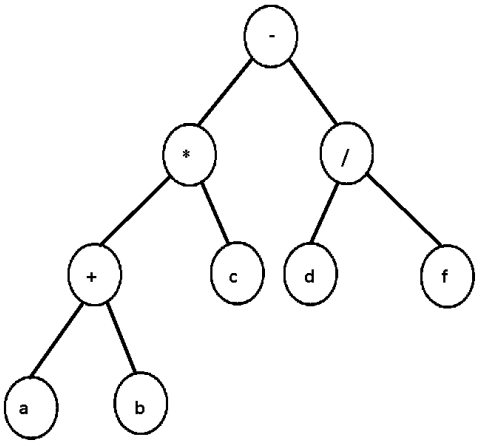

Figure 1.4 Tree of arithmetic expression

Go around the tree constructed in preorder manner, which means that the sign of the operation was recorded at the root of the tree (subtree) precedes its operands:

-*+abc/df and obtain the so-called prefix (Polish) notation of arithmetic expressions.

Bypassing the same tree in the postorder manner, that means the operands preceded by an operation, that gives opportunity to write an arithmetic expression in the so-called postfix (reverse Polish) notation, these expression: **ab+c*df/-**

Again, that when writing arithmetic expressions in prefix and postfix form, there is no need to use brackets because each of these forms of records are of the original expression, this expression can be completely restored with all its brackets.

Symmetric (inorder) traversal of the tree means that the first pass left direct descendant of the root of the subtree, then the root contains the sign of operation taking process and then traversed right direct descendant of the same root. This option allows bypassing the so-called infix notation of expression: **a+b*c-d/f** which the original expression, initially containing the brackets cannot be recovered uniquely, and this requires further analysis. Note that, as a rule, especially traversal determined by the nature of these problems. In additional to examples already discussed here can also refer to the examples of computer graphics, where the challenge of removing hidden surfaces, bypassing the 2-dimensional binary tree, spatial division was organized in order of visibility of line segments with respect to the observation point. There will also be relevance to recall that under translation programs, for coding of arithmetic expressions (and not only) used postfix form, i.e. already mentioned reverse polish writing. “Stack” is used to calculate the values of arithmetic expression.

1.1.3 Representing of binary trees

Representing of binary trees. One way is to try to imagine the tree as an array of conventional structure. In this connection, we consider two examples of the tree.

The first tree can be taken as a complete or almost complete, like in figure

1.5, from the left tree. Such a tree has a height $log_2 n$ and representation as an array is very compact.

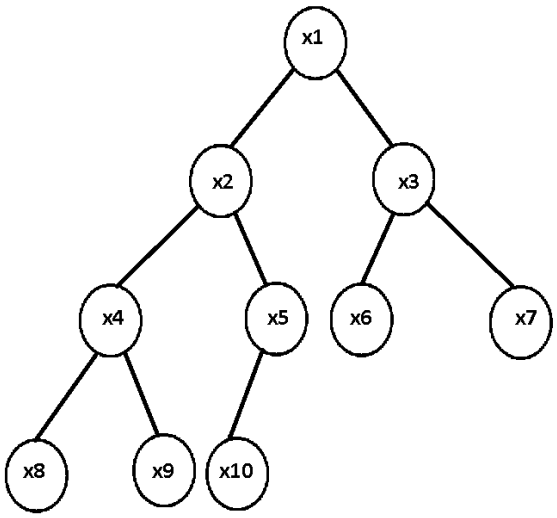

Figure 1.5 Almost complete tree

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10

Figure 1.6 Allocation in memory

The second tree has height n and demonstrates the opposite properties of this type representation of trees.

An example is shown in Figure 1.7:

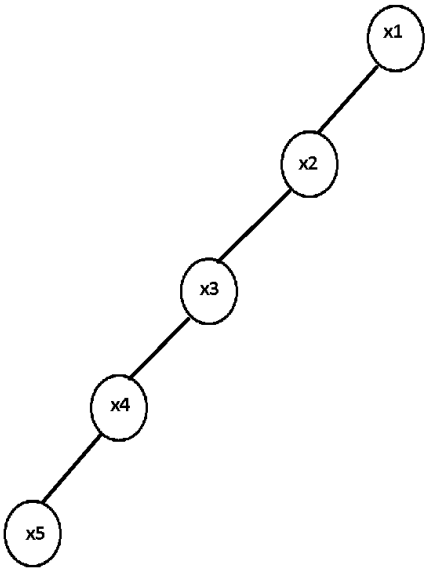

Figure 1.7 Tree with height n

x1	x2	-	x3	-	-	-	x4	-	-	-	-	-	-	-	-	x5

Figure 1.8 Allocation in memory

Requiring much more memory than is actually needed.

Thus, it becomes clear that in order to practically apply this method of representation to any tree it is necessary to complement this tree to a complete or almost complete from the left tree. It is obvious that, in the array representing the tree will get emptiness, where in the tree false vertexes are appeared. The closer the initial tree, to the tree consisting of only one branch (path), the less effective shows the given way of representing trees.

1.2 Complete tree

The vertex of a binary tree is called a growth point of this tree, if vacant direction can attach a new edge, i.e. new edge will emanate from this point(vertex), like shown in Figure 1.9:

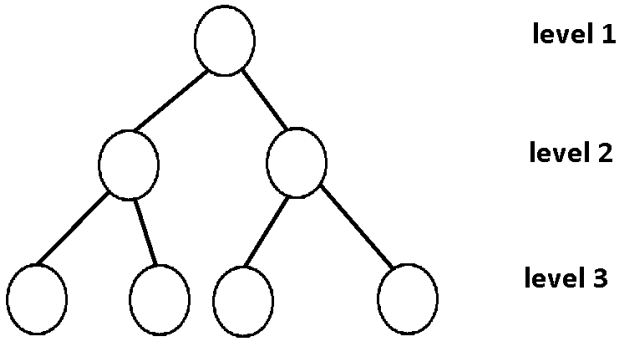

Figure 1.9 Growth point

Binary tree is called complete tree if all its growth points are at the last level. Each vertex of the complete tree, but the final ones has exactly two direct descendants, and all paths starting at the root and ending at a final node (leaf) have the same length. Recall that in this case each edge on this path is traversed only once.

It is easy to calculate that the number of vertices on the k-th level in the full(complete) binary tree is 2^{k-1}, k≥1.

The number of vertices n in the complete binary tree the maximum level which is equal to m≥1, obviously is equal to

$$\sum_{k=1}^{n} 2^{k-1} = 2^m - 1 \qquad (1.1)$$

Search for an item in the full tree is carried out very quickly, not more than for $log_2 n$ comparisons, but hope that the original array has the number of element equal to "power of two minus one" is not very large. In this connection, consider the notion of an almost complete tree, a tree of this type is also called as leveled, which you can build for any array containing an arbitrary number of elements.

1.3 Almost complete tree

Binary tree is called an almost complete, if all of its growth points are located only at the last and penultimate levels of the tree. Sometimes this tree is called also fully balanced or balanced tree.

It is easy to show that if an almost complete tree contains vertices of level at most m, then the maximum number of comparisons required searching for an element is evaluated

$$log_2 n \ < \mathrm{m} \geq log_2 n + 1 \qquad (1.2)$$

i.e. task of searching for an almost complete tree is solved practically for the optimal minimum-time, i.e. for the minimum number of comparisons. Algorithm for constructing almost complete tree is simple, but there is also a major drawback of such trees. If the array is static, i.e. it does not change either the number of elements or the elements themselves, the use of almost complete tree search is justified. Once spent time on the organization of such a tree, later solved only the problem of searching, and very effectively. Another thing, if the array elements are periodically changed or they added or removed. Consider the traditional case where the array consists of 6 elements and an almost complete tree is Figure 1.10:

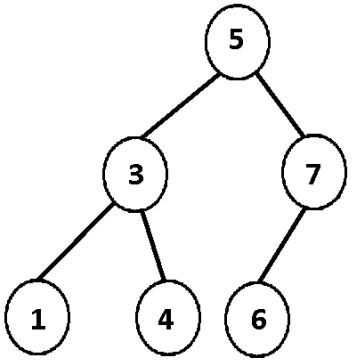

Figure 1.10

Adding only one element "2" may lead to the almost total restructuring of the tree in which only one element of the array remained in its original location place (Figure1.11):

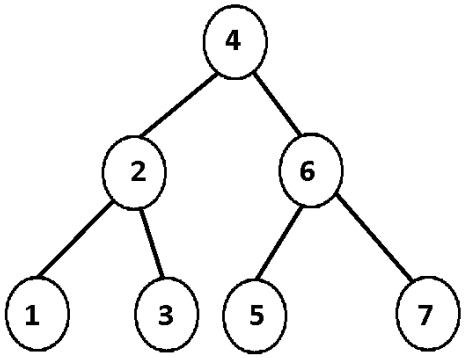

Figure 1.11

Despite the fact that the algorithm for constructing almost complete tree from the original array is simple, the effectiveness of using trees for this type for dynamic arrays can be quite low because, as we have seen when a small change in the array took place it may require a substantial or even a complete restructuring (overhaul) of the tree.

1.4 AVL trees

AVL trees. Although, this class of binary trees has quite complicated compared to the previous ones the algorithm of construction, but it also has important and positive features (Figure 1.12).

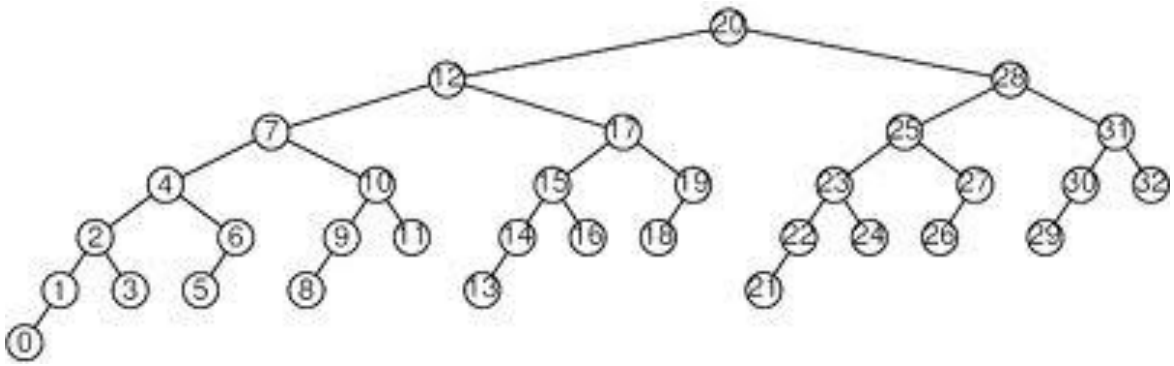

Figure 1.12 AVL tree

First, in spite of the fact that trees of this type may look non symmetric but they have, as it turns out, not long branches as in principle, one can guess carefully reading the following definition. The maximum length of the branches emanating from the root of this tree does not exceed values c* $log_2 n$, where c≈1.44. Secondly, the addition or removal of elements (vertices) does not require a restructuring of the entire tree, while, strange as it seems at first glance, the algorithm for constructing such a tree, in spite of some difficulties in the justification, is not hard in essence. AVL-trees are often called balanced trees.

Length branch emanating from the vertex V of a tree is the difference between the maximum level of vertices belonging to this branch and the level of the

vertex V.

It is clear that the length of the branch is the length of the longest path belonging to this branch.

The tree is called balanced (AVL) tree if for each vertex the tree absolute value of the balance does not exceed 1.

1.5 B-tree

B-tree. This method uses the concept of a binary search tree, but modifies this concept so that the tree is paged. Therefore, B-trees are often referred to as trees paged media. Suppose that we want to organize an in-memory search tree with a very large number of vertices (millions and more). Let the free-OP is not sufficient for the simultaneous storage of all vertices at once. Part (though very significant) peaks should remain on the disk. There arises the problem of the organization of the tree to be able to read and process the right group of vertices. Such a group of vertices is called "page". Thus, all the extra-large search tree is divided into a number of pages and reading of the vertices with the drive is performed by page. This reduces the number of calls to external memory (Figure 1.13).

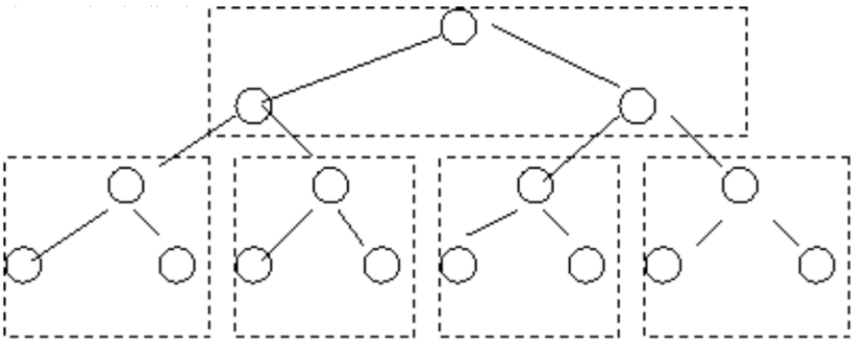

Figure 1.13 Paging of tree

For efficient paging of extra large search tree, the very important condition is uniformity of the growth, maximal conformity of perfect balance. Unfortunately, the condition of perfect balance leads to a very complex algorithms, so Bayer introduced the concept of a pageable tree (B-tree).

B-tree of order m with paged organization - is a data structure that satisfies the following conditions:

1. The entire set of n elements is divided into separate pages, and each page may contain from m to 2m vertices except for the root of page with the number of nodes varying from 1 to 2m
2. Each page is either a terminal (no children) or has exactly one more child than the current number of vertices of the page
3. All terminal pages are on the same level

Example(Figure 1.14). We consider the simplest B-tree of order m = 2 with keys of type integer. In accordance with the above rules, each page of the tree contains from 2 to 4 vertices, and the root - from 1 to 4. Each a non-terminal page can have 3 to 5 children.

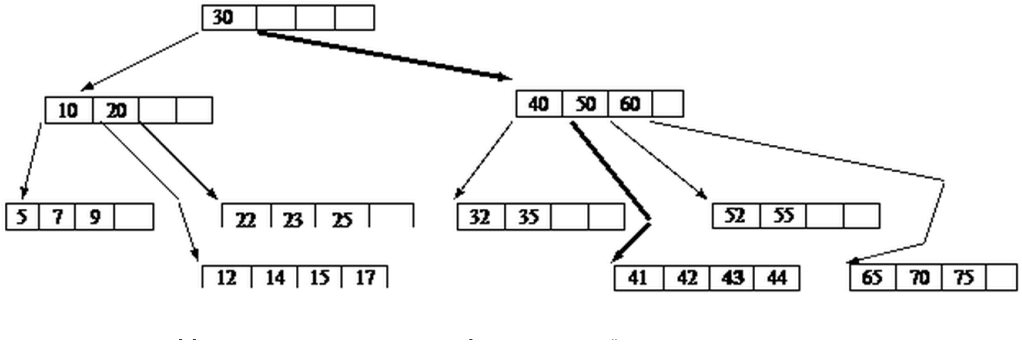

Figure 1.14 B-tree

Since the B-tree is a generalization of the classical search tree, it has the following properties:

1. On each page the elements located in ascending order of keys
2. Each page contains a descendant keys at a specific range, which is defined by the parent key

The last property is a generalization of the basic rules of the search tree.

The effectiveness of the use of B-tree to find the keys in the ultra-large data sets can be estimated as follows. If the B-tree is of the order m, the number of elements - n, then the worst case is thus the case where each page is the minimum possible number of components, namely - m. In this case the height of the B-tree is determined by the $log_m n$, namely height and determines the number of calls to external memory.

1.5.1 B-tree as a data structure

The basic element of the B-tree is the page, because it contains all the basic information. Therefore, it is first necessary to describe the structure of the page of B-tree.

Each page should contain the following information:

1. the current number of elements in the page (it varies from m to 2m)
2. a pointer to the page, which is the most left child of this page
3. the bulk of the elements of a page, the array dimensions 2m, every element is a record with the following fields:
    1. key of some type
    2. pointer to a page which is a descendant of the current page containing the keys, bigger than key of this item
    3. informational part (like some structure or a pointer to a memory location)

It can be seen that the B-tree is a difficult combination structure, introducing a set of pointers to related records. To use the B-tree, a standard set of operations is needed the search of a given element, adding an item, remove an item.

1.6 About programming environment

C # is object-oriented programming language. Developed in 1998-2001, a group of engineers led by Anders Hejlsberg at Microsoft as a language for developing applications for the platform Microsoft. NET Framework and later was standardized as ECMA-334 and ISO / IEC 23270[1].

C # refers to a C language family's syntax, the syntax of which is closest to the C + + and Java. The language has static typing, supports polymorphism, overloading of operators (including operators of explicit and implicit type), delegates, attributes, events, properties, and methods of generics, iterators, anonymous functions with support of closures, LINQ, exceptions, comments in the format XML[2].

C # adopted many features of their predecessors - the languages C + +, Java, Delphi, Modula and Smalltalk, based on the practice of their use, eliminates some models which have proven to be problematic in the development of software systems, etc, C #, unlike C ++ does not support multiple inheritance classes (in the meantime allowed multiple inheritance of interfaces).

1.7 Why C#?

For the implementation of the software high-level programming language is used C # and development environment - MS Visual Studio 2008. This choice is due to the following criteria:

1. The development environment and language allow to quickly and easily create high-level user interface, using the forms designer, minimizing the work of the programmer. An alternative could be considered a development environment Code Gear 2009 but MS Visual Studio 2008 clearly has a number of advantages as a natural platform for the development of a Windows.
2. C # language also has a number of advantages compared to Object Pascal, since C# is newer and more advanced language.
3. A very high level of security mechanisms of code is implemented in it.
4. Another definite plus is the presence of Russian-developed referral system, which greatly simplifies the process of programming.
5. Another alternative language might be JAVA, but it requires work to make their own Framework, while necessary Framework for programs is written using C # which is already present in the operating system.
6. Another advantage is the ability to use both managed and unmanaged code, both manual and automatic memory management. Major competitors JAVA and Object Pascal does not allow simultaneous use of both of these models[3].

7. Another advantage may be using the byte-code, which provides high performance and better utilization of the hardware capabilities of the PC.

As an alternative, Web Development Environment is not considered, because obtained a single-user application does not presuppose the existence and use of the web server and scripting languages.

2 NON-NUMERIC ALGORITHMS

2.1 Used tools

2.1.1 Used libraries

The System.Collections.Generic namespace contains interfaces and classes that define generic collections, which allow users to create strongly typed collections that provide improved performance and security types, compared with non universal strongly typed collections [4].

The System.Drawing namespace provides access to the functionality of the GUI GDI +. Namespaces System.Drawing.Drawing2D, System.Drawing.Imaging, and System.Drawing.Text provide additional functionality [4].

The Graphics class provides methods for drawing on the display device. Classes such as Rectangle and Point, encapsulate elements of GDI +. The Pen class is used to draw lines and curves, and the classes derived from the abstract class Brush, used to fill shapes [4].

2.1.2 Used built-in classes

The Graphics class provides methods to display the objects in the display device. Graphics Object is associated with a particular device context [5].

The bitmap contains data display elements (pixels) and the attributes of a graphic image. There are many standard formats to save the bitmap to a file. Class GDI + supports the following file formats: BMP, GIF, EXIF, JPG, PNG and TIFF. You can create images from files, streams and other sources, using one of the constructors Bitmap, and keep them in a stream or a file system by using the Save. Images are drawn on the screen or in memory using the DrawImage method of the Graphics [5].

Pen object draws a line of specified width and the specified style. DashStyle property is used to draw a variety of dot-dashed lines. Drawn with this Pen line can be filled using a different fill styles, including solid colors and textures. Depending on the fill style, brush or textures selected object are required [5].

2.2 Algorithm of the project

Algorithm of this work is non-numeric, so it is impossible represented as formula. When constructing the tree, the root of the tree can be associated with a vertical line, and his direct descendants are mapped horizontal lines passing through these vertices. Direct descendants of the vertex are placed on the next level of the tree and they are again mapped to vertical lines passing through these vertices. Each line l_x passing through a fixed vertex x divides the remainder of the array into two parts: the vertices belonging to the left branch L_x emanating from x and the vertices belonging to R_x, also emanating from the vertex x. In the figure, these two parts will be located on both sides of the vertical (horizontal) line l [6].

The first example is an example of a tree constructed from the array a,d,I,b,h,g,f,e,c with the location of points a,b,c,d,e,f,g,k,i on the plane are fixed in advance. (figure 2.1)

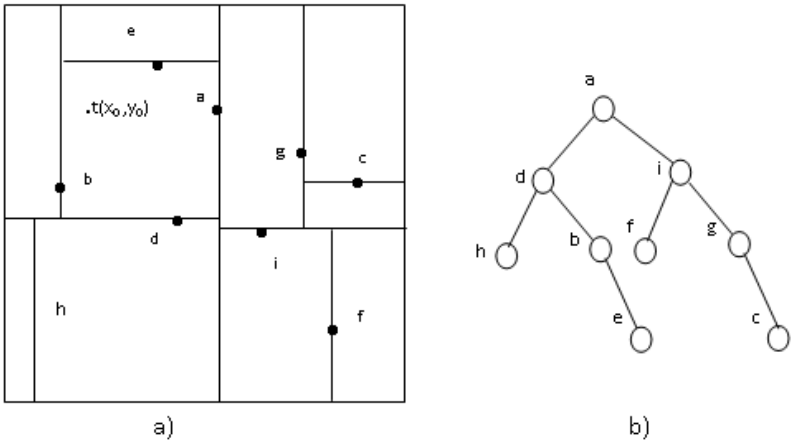

Figure 2.1 a) the partition of the plane by cut off straight line-grid, b)corresponding to 2-d tree.

The fact that one on Figure 2.1. b) The tree is almost complete, as if by accident that the random arrangement of the array elements relative to each other. These random coincidences easily see adding a new node, the figure is a vertex t with coordinates x0, y0.

For greater contrast with the situation depicted in Figure 2.1. Assume that the array elements act as follows: c, f, g, e, i, h, b, d, a. We construct the grid and the tree, starting from the top and to bottom:

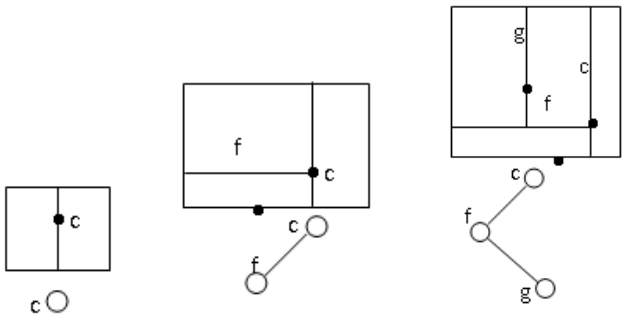

Figure 2.2

The next steps, acting in the same spirit, we come to the final configuration of the location of points and lines:

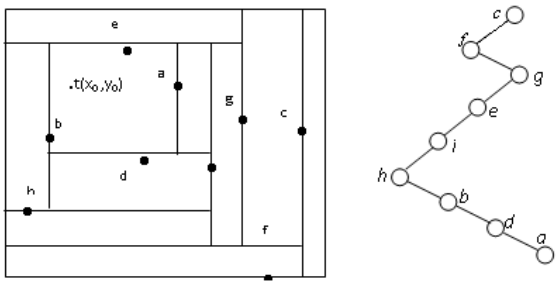

Figure 2.3.Net and 2-d-tree for the array c, f, g, e, i, h, b, d, a

It is not difficult to understand how effective the process of localization of t in the case shown in Figure 2.3. Location tops 2-d-tree in Figure 2.3 clearly illustrates this. Localization of t on the grid one needs to understand the algorithm that determines the minimum rectangle that contains the point t. Basically, in general, the 4th point defining the rectangle is required to determine. For the situation depicted in Figure 2.3 it will be the point of a, d, b, e[7].

If the above examples, and questions remain about the comments made by the construction of the grid and 2-d-tree, then we can recommend, first selecting the scale of one division, attach a numeric axis (abscissa and ordinate) on the rectangle of the image, considering the usual horizontal axis, the x-axis, and the vertical axis, y-axis. The following is offered instead of the character of each tree node to record its coordinates, for example, (Xe, ye) instead of e, and each edge of the 2-d-tree attribute "weight"-variable x or y, depending on which variable is then compared .

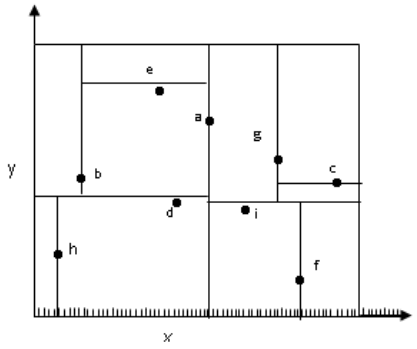

Figure 2.4

In addition, until the situation becomes pretty clear, it would be helpful if one builds a grid and a tree for every point in the plane rectangle as a subscript to write symbol ancestor of the summit. Then, for example, letter h (Figure 2.4) takes a form hd, but instead will appear g gi, such that the final picture will be the

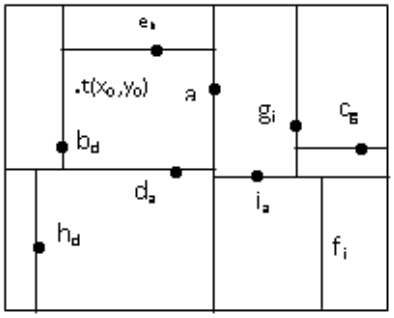

Figure 2.5

It is clear that the practice of the 2-d tree for solving the problem of localization point, found a "square." To the point t (Figure 2.5) there can be a fairly large portion of the image. Although not a final, this piece will give too large error localization point t, failing to provide the required accuracy. In this regard, it can be assumed that the considered square (rectangle) itself is an independent, like the original, but a much smaller scale, the image which has its own grid.

It is reasonable to assume that such investments can be a fixed amount which is sufficient for what would eventually get a reasonably accurate localization (Figure 2.6).

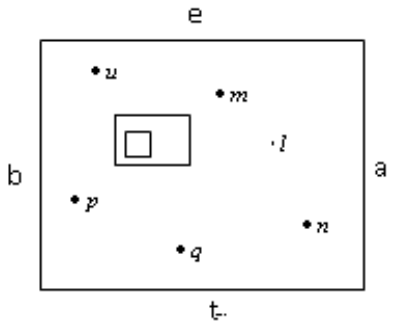

Figure 2.6 Localization of the point

With this approach, we can assume that for the localization of the "point" it is not necessary to do just enough time-consuming scan of the image, focusing on the very small scale of the grid. At the first stage the relatively large fragment, and then, if necessary finer etc. until an acceptable accuracy of localization.

The sequence of steps for constructing and / or addition of a new node will be "a little bit" to resemble the work of satellite retrieval, information system. For this purpose, the so-called quadrant trees [8]. Here we mention that to solve such a problem can be applied to the hash function.

To improve the efficiency of practical use it is desirable that 2_d tree would be complete or almost complete tree, in connection with which one wants to sort the original array to the coordinates x and y, respectively. At each step of the construction of a grid as horizontal and vertical lines to divide the rest of the array roughly equal.

However, we already know that for dynamic arrays efficiency is almost complete tree falls markedly. The need to add even a single element (point) may require a re-build the entire grid.

Additionally, one may notice that the meshing algorithm and 2-d-tree, in principle, can be applied not only to the rectangular areas, for example, and for a convex domain with a given boundary (Figure8.12 a). Feature points m, j, k, l to the array a, d, i, b, h, g, f, e, c generates the same sequence as that for the rectangular region. However, addition of n points has increased the number of non-standard forms of non-rectangular region

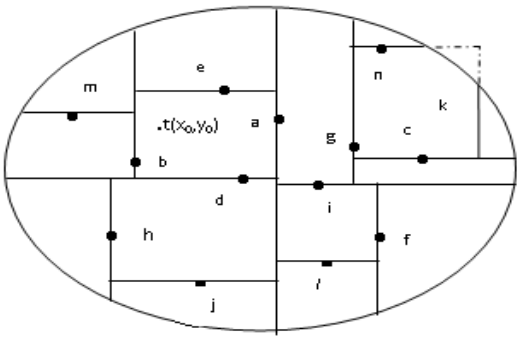

Figure 2.7

Next, we present a program of constructing a grid and the corresponding 2-d-tree for a random array of points defined by their coordinates [9]. The parameters of the basic procedure is itself the source array of points and the points defining a rectangular area. The basic procedure invokes the procedure **Octangle** is the procedure of constructing a rectangular area on a given angular points. Besides the main body of the procedure contains a reference to the procedure of drawing a line of horizontal and vertical lines, which depends on the left, right, bottom, and top boundaries of the coordinates xi, yi of the current point, respectively. It is clear that these boundaries for each line are under construction and should be defined in the calculations in the main proceedings. Wood is in the form of slots linked list with an element (Table 2.1):

Table 2.1

Class variables

left_des	**x**	**y**	**right_des**	**anc**

Which is accessed by sequentially passing tree link v, starting with the root and further to the right along the top branch. A new item with the same structure, if necessary, is created using the procedure **next_record**, returns a reference to this new element.

Under the covers, **tree** useful functions:

$$\begin{cases} z(a,b)= \begin{cases} a\ , \text{if } z=0 \\ b, \text{if } z=1 \end{cases} \end{cases} \quad \text{and} \quad \begin{cases} z.v= \begin{cases} x.v\ , \text{if } z=0 \\ y.v\ , \text{if } z=1 \end{cases} \end{cases}$$

It just uses an auxiliary variable Bound (Figure 2.8),

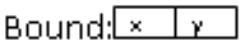

Figure 2.8 variable “Bound”

consisting of two real fields and real variable **border**.

There are also entries such as: x.anc (v), their interpretation somewhat unconventional. It is proposed to start reading from the innermost structure, ie c v and further, in the course of computing, moving from *left to right*, the last field is allocated x. At the same time, what if either the internal construction will nil, the need for further calculations and not all such constructions are defined as nil. Isolation, in the computing field x and y for the variable **Bound**, isalso carried out by the variable z. Code of the procedure is as follows:

```
procedure 2_d_tree(a,n,x0,y0,x1,y1);
begin
   temp←nil;
   LBD:=x0;
   RBD:=x1;
   UBD:=y1;
   BBD:=y0;                                  //* believe that
   rf_root← nil;
   Rectangle(x0,y0,x1,y1);  //* x0 < x1, y0< y1, draw a rectangle //
   for i=1 to n do  begin0
       z:=0;
       x.Bound1:=RBD;
       y.Bound1:=UBD;
       x.Bound2:=LBD;
       y.Bound2:=BBD;
       tree(rf_root,x[i],y[i]);
       procedure tree(v,a,b);   begin1
            while  v≠nil do
             begin2 temp← v;
                 if  z(a,b) ≤  z.v
                     then begin z.Bound1:=z.v; v← left _des(v) end
                  else
                       begin z.Bound2:=z.v; v ←rigt _des(v); end;
                   z:= 1- z;
                end2;
                z:= 1- z;  //* restore the desired value of the variable z //
                v←next_record;
                x.v:=a;
```

```
                y.v:=b;    //* fill in the fields of the new element //
                left_des(v) ←nil;
                rigt_des(v) ←nil;
                anc(v) ← temp;

                if anc(v)=nil
                then begin rf_root←v; line(BBD, UBD, x[1],π/2); end;
                        //* asked root, drew a vertical line //
                else
                begin3 inf_anc(v)←v;
                //* write in reference to the corresponding field ancestor v//
                    if  z.v >  z.anc(v)
                    then border:=z.Bound1
                    else border:=z.Bound2;
                    if z=0
                    then     line(x.anc(v), border,y[i], 0)   //* by yi draw a
                    horizontal the line//
                    else
                       line(y.anc(v), border, x[i], π/2); //*z=1,by xi draw a
                       vertical the line//
                end3;
            end1;
        end0;
    end; / / * This assignment operator to link it ← //
```

The final step of constructing the line segment to a fixed point in the standard procedure line, in the case of a horizontal line, one of its boundaries is x.anc (v), a second boundary border. In the case of the vertical segment is similar: the boundaries of the segment passing through the point xi, yi is y.anc (v), border. The last parameter of the procedure is the angle line with respect to the horizontal axis, which is followed by another paint line [10].

The main purpose of the procedure tree, this computation the variable z, which in turn allows the outside of the body of procedure tree, calculate zv, z.anc (v), z.Bound1, z.Bound2. Recorded in the process tree inequality zv ≤ z (a, b), as in any other value, where it participates variable z, and entered into the text for clarity and convenience of inspection, a corresponding relationship between the variables x and separately for y . For this inequality zv ≤ z (a, b) it does not matter on which, namely, the variable x and y it holds, on it is essential that, if it is satisfied, the next step should be to the right direction. In addition, for what would be in the body of the procedure tree correctly perform the operation inf_anc (v) = v, Ie the last "non-trivial" step of this procedure, you will need to figure out which field of the left or right child at anc (v) we need to write a reference to the newly inserted element. To do this, we must recall, which variable x or y was to comparing, which led to the formation of the last element. In this task, we can help the variable z, ie its last value, adjusted

after the release of the procedure tree. If the value z = 1, then compare the obviously necessary to y, otherwise by x.

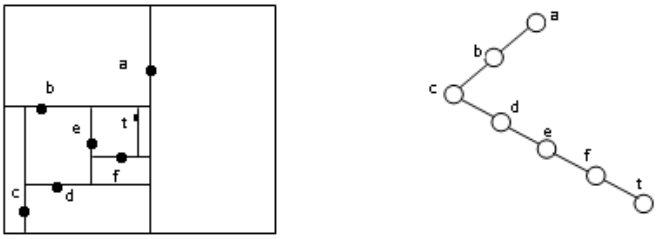

Figure 2.9 Grid and tree of example

	0.bound1	1.bound1	0bound2	1bound2	border
a, i = 1	RBD	UBD	LBD	BBD	-
b, i = 2	x_1	...	...	...	LBD
c, i = 3	x_1	y_2	...	...	BBD
d, i = 4	x_1	y_2	x_5	...	x_1
e, i = 5	x_1	y_2	x_5	y_4	y_2
f, i = 6	x_1	y_2	x_5, x_3	y_4	x_1
t, i = 7					

Figure 2.10 Illustration of tree and grid as a table

Note that, in spite of its name, the procedure 2_d_tree builds only the net, but a slight modification of this procedure allows us to construct this grid and the corresponding tree.

Here we note that the recursive procedure for constructing the array is almost complete 2_d-tree, it may be even simpler than the above 2_d_tree procedure(Figure 2.11).

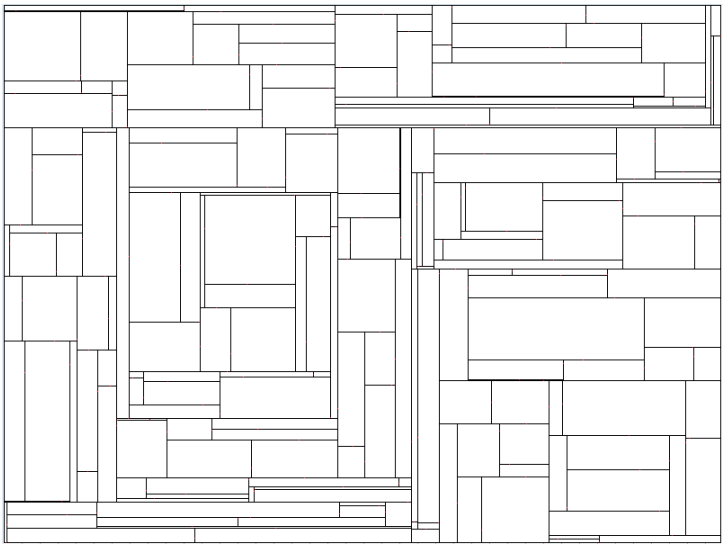

Figure 2.11

At the end of the fragment, we note that even more interesting object to the 2-d tree is the so-called 3-d tree, or even better would be to mention the k-d tree design features which can be drawn in the literature and in this manual are not considered [11].

3 REALIZATION OF THE ALGORITHM OF SPATIAL SEARCH

This section will provide the information about how algorithm work and how it was implemented.

3.1 Description of the use case diagram

A *use case diagram* at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well [12].

3.1.1 Use case diagram of the project

A user who needs help sends a request to Relief Services[13]. Then the request processed by the server, which means program localized the request's point to the minimum rectangle. After determining the location, the server sends a request for assistance and they've sent services to help the user (Figure 3.1).

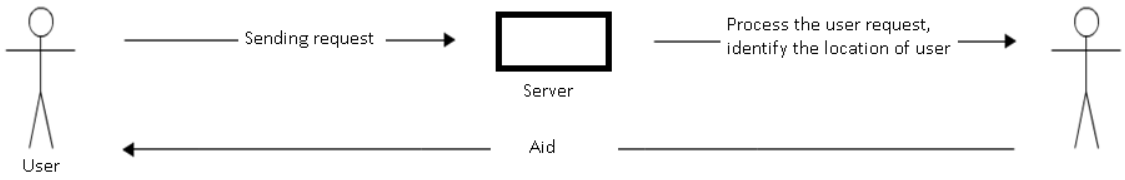

Figure 3.1

3.2 Model of project

The figure below shows the model of spatial search.

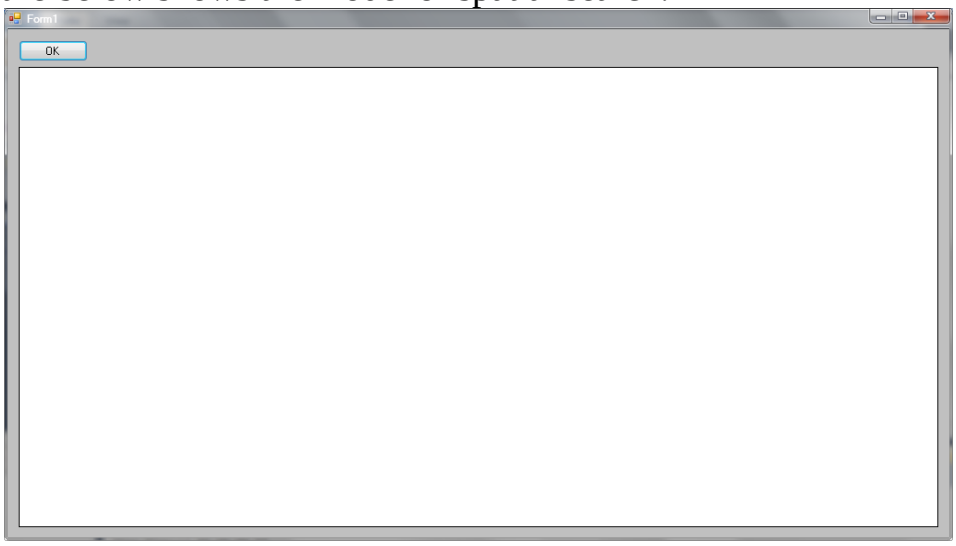

Figure 3.2

To run the model button "OK" should be pressed.

After clicking the button, red dot will appear. Red dot means the location of the request on the map.

Figure 3.3

Then tree is constructed and appropriate grid. The figure below shows the localization of the point. Localized rectangle is red.

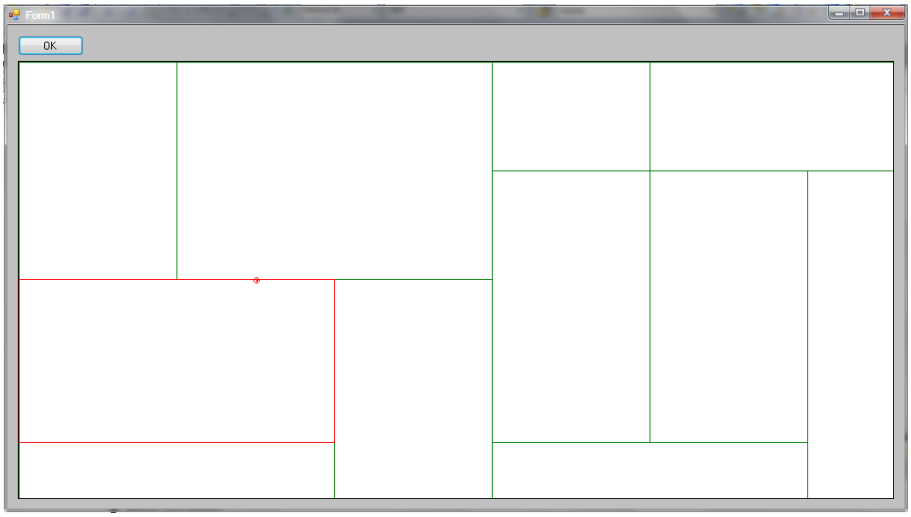

Figure 3.4

As we can see in the figure below, this localization has too big infelicity, so it is desirable to repeat this operation. This operation will be repeated only within red rectangle.

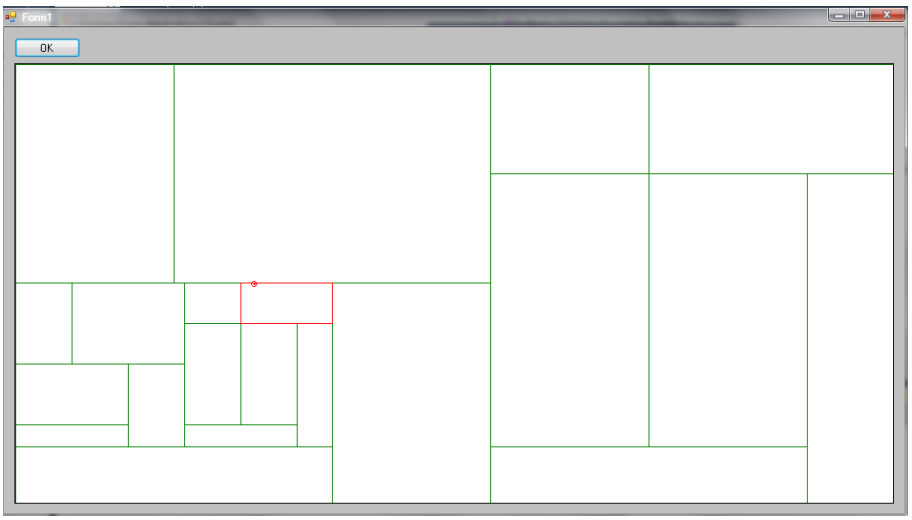

Figure 3.5

Infelicity is still considerable.

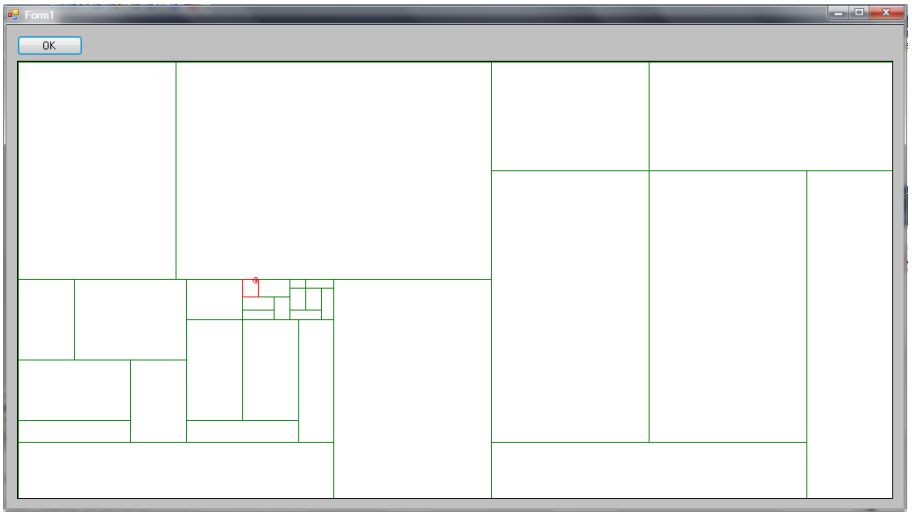

Figure 3.6

One should recursively repeat until it gets the desired result.

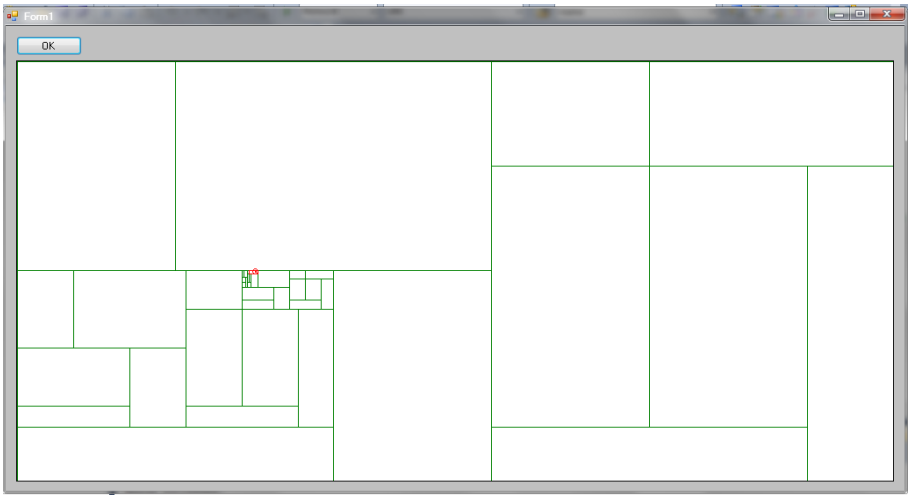

Figure 3.7

After repeating this method recursively, we finally obtain acceptable accuracy.

3.3 Realization of project

Let's go directly to the project realization.

First of all we create two classes. The first class is "Coordinates", this class stores a value of x and y. The second class is "Node", it stores an element of class of "Coordinates", that is, the values of x and y of given vertex, and reference for the parents and the references for the children (if they exist).

```
private class Coordinates
{
    public int x;
    public int y;

    public Coordinates(int x, int y){
        this.x = x;
        this.y = y;
    }
}

private class Node
{
    public int left_des;
    public int right_des;
    public int anc;
    public   Coordinates a;

    public Node(Coordinates a){
        this.a = a;
        anc = -1;
    }

}
```

Listing 3.1

Then an array of the class "Coordinates" is created (in this project, they are taken randomly).

```
Coordinates[] a = {new Coordinates(6,5),new Coordinates(3,4),new Coordinates(9,2),
                   new Coordinates(4,6),new Coordinates(10,5),new Coordinates(8,1),new Coordinates(2,7),
                   new Coordinates(8,7),new Coordinates(2,2),new Coordinates(8,5)};
```

Listing 3.2

This array is passed to the function "two_d_tree". The id of parent (if it exist) and the value of z is identified in this function. z is a variable that contains the value 1 or 0, ie, it determines the direction where the line will be drawn in horizontally or vertically. Also there are variables "BBD", "UBD", "LBD", "RBD". These variables are responsible for the start position of x and y, "LBD" and "UBD" respectively. And for end position of x and y, "RBD" and "BBD" respectively. The values of variables vary according to x and y of the parent node. All this data is passed to the function

"tree".

```
private void two_d_tree(Coordinates[] a, int n, int x0, int y0, int x1, int y1)
{
    kx = (x1-x0)/11;
    int startX = LBD, startY = BBD;
    ky = (y1 - y0) / 8;
    space.DrawRectangle(greenPen, x0, y0, x1 - x0, y1 - y0);
    Point[] points = {};
    //space.DrawCurve(greenPen,new Point(),);
    int rf_root_i = -1;
    z = 0;
    for (int i = 0; i < n; i++)
    {
        LBD = x0;
        RBD = x1;
        UBD = y1;
        BBD = y0;
        z = 0;
        for (int j = 0; j < i; j++)
        {
            rf_root_i = j;

            if (z == 0)
            {
                z = 1 - z;
                if (a[i].x > nodes[j].a.x)
                {
                    LBD = nodes[j].a.x*kx+startX;
                    if (nodes[j].right_des < 0)
                        break;
                    else
                        j = nodes[j].right_des-1;

                }
                else
                {
                    RBD = nodes[j].a.x*kx+startX;
                    if (nodes[j].left_des < 0)
                        break;
                    else
                        j = nodes[j].left_des-1;
                }

            }
            else
            {
                z = 1 - z;
                if (a[i].y > nodes[j].a.y)
                {
                    BBD = nodes[j].a.y*ky+startY;
                    if (nodes[j].right_des < 0)
                        break;
                    else
                        j = nodes[j].right_des-1;
                }
                else
                {
                    UBD = nodes[j].a.y*ky+startY;
                    if (nodes[j].left_des < 0)
                        break;
                    else
                        j = nodes[j].left_des-1;
                }
            }
        }
        nodes[i] = tree(a[i].x, a[i].y, i, rf_root_i, startX, startY);
    }

    LBD = x0;
    RBD = x1;
    UBD = y1;
    BBD = y0;
    bounds(search, nodes, startX, startY);
}
```

Listing 3.3

Grid lines are drawn in this function. The first step is to check if the node has an ancestor, if not then a rectangle and the function ends should be drawn, because this is the first node of the tree [14].

If a node has an ancestor, then the value of z is checked, if the value of z is 0, then the value of y is checked, if y of current node is bigger than y of ancestor, then id of node will be recorded in the " right_des " property of ancestor and the beginning of the line will start with y of ancestor, if y of current node is smaller than the y of ancestor, then id of node will be in the " left_des" property of ancestor and the end of the line is the y of ancestor, then a horizontal line is drawn[15].

If the value of z is 1, then the value of x is checked, if x of current node is bigger than x of ancestor, then id of node will be recorded in the " right_des " property of ancestor and the beginning of the line will start with x of ancestor, if x of current node is smaller than x of ancestor, then id of node will be in the " left_des" property of ancestor and the end of the line is the x of ancestor, then a vertical line is drawn.

```
private Node tree(int x,int y, int i, int j, int startX, int startY)
{

    int zBound1;
    int zBound2;
    Node temp1 = new Node(new Coordinates(x, y));

    if (j == -1)
    {
        rf_root = temp1;
        space.DrawLine(greenPen, x * kx + startX, BBD, x * kx + startX, UBD);
    }
    else
    {
        if (nodes[j] != null && j>=0)
        {
            temp = nodes[j];

            if (z == 0)
            {
                zBound1 = UBD;
                zBound2 = BBD;

                if (y <= nodes[j].a.y)
                {
                    zBound1 = nodes[j].a.y*ky+startY;
                    nodes[j].left_des = i;
                }
```

```
                else
                {
                    zBound2 = nodes[j].a.y*ky+startY;
                    nodes[j].right_des = i;
                }
                space.DrawLine(greenPen, x * kx+startX, zBound1 , x * kx +startX, zBound2 );
            }
            else if (z == 1)
            {
                zBound1 = RBD;
                zBound2 = LBD;

                if (x <= nodes[j].a.x)
                {
                    zBound1 = nodes[j].a.x*kx+startX;
                    nodes[j].left_des = i;
                }
                else
                {
                    zBound2 = nodes[j].a.x*kx+startX;
                    nodes[j].right_des = i;
                }
                space.DrawLine(greenPen, zBound2, y * ky +startY, zBound1 , y * ky +startY);
            }
        }
    }

    //v=temp1;
    temp1.left_des = -1;
    temp1.right_des = -1;
    temp1.anc = j;

    return temp1;
}
```

Listing 3.4

These two functions will be repeated for each element of the array "a", until the tree will fully built and grid will be painted [16].

After the first construction of the tree and the grid, the result can have to large infelicity, so we need to repeat this function multiple times. To do this, we need to determine the value of boundary of localized rectangle.

To do this, function “bound” is needed.

This function takes the value of a call, and then passes through the elements of the tree and then identifies the boundary of localized rectangle, which means the new values of "LBD", "RBD", "UBD", BBD ".

```
private void bounds(Coordinates b,Node[] a, int startX, int startY)
{
    for (int i = 0; i < a.Count(); i++)
    {
        if (z == 0)
        {
            if (startX+a[i].a.x*kx > b.x*1000/11)
            {
                RBD = a[i].a.x * kx+startX;
                i = a[i].left_des - 1;
            }
            else
            {
                LBD = a[i].a.x * kx+startX;
                i = a[i].right_des - 1;
            }
        }
        else
        {
            if (startY+a[i].a.y*ky > b.y*500/8)
            {
                UBD = a[i].a.y * ky+startY;
                i = a[i].left_des - 1;
            }
            else
            {
                BBD = a[i].a.y * ky+startY;
                i = a[i].right_des - 1;
            }
        }
        z = 1 - z;
        if (i < 0)
        {
            break;
        }
    }
    space.DrawRectangle(new Pen(Color.Red,1), LBD, BBD, RBD-LBD, UBD-BBD);
}
```

Listing 3.5

After the end of the function "bounds", function "tree" and "two_d_tree" are repeated with the new values "LBD", "RBD", "UBD", BBD ".

4. ECONOMIC EFFICIENCY OF THE PROJECT

When writing a graduation project we must not forget to optimize spatial search to calculate the costs of implementation of the project, to assess the project through an analysis of its economic performance and to draw conclusions about its feasibility.

This assumes that the optimization of the spatial search will be used in the field of navigation for support services.

The software product should be designed so that it performs its functions without the expense of resources (RAM, CPU time, and others - on the stage of the operation, development time and financial resources - at the stage of use of the software product) [17].

4.1 Economic efficiency

Economic efficiency - means to get the maximum possible benefits from the available resources. To do this, one should constantly correlate benefits and costs or, to put it another way, to behave rationally. Rational behavior is when the producer and consumer seek to get the highest efficiency and to maximize this benefit and minimize costs.

One of the important parts of the efficiency of the economic system is the efficiency of capital investments. It is expressed as the ratio of the obtained effect to the capital investments that caused this effect. The effectiveness of capital investments measured by set of indicators, includes the overall effect of capital investment, the rate of return, payback period, the comparative effectiveness and other indicators of economic efficiency of capital investments are used to compare alternative investment projects and the choice of optimal design [18].

A rather large role in the economic process (ie, whether software economical or not) depends on the programmers who are creating this software. Then play the role of the creation, debugging, and operation of the program.

For the economic analysis of the effectiveness of the investment project indicators such as the payback period (PP) are traditionally used.

Payback period (PP) is the time during which income from investments is equal to the initial investment (i.e. the period is necessary to ensure that the funds invested in the project are fully returnable.)

Together with the net present value (NPV) and internal rate of return (IRR) a tool for evaluating investments is used.

Payback period - is an excellent indicator that provides one with a simple way to know how long does it take the customer to recover the initial cost. This is particularly important for businesses located in countries with fragile financial system, or business associated with advanced technology, where rapid product obsolescence is the norm, which makes a quick refund of investment costs in an important problem. Just these figures tell us about how to effectively apply the software product taken by us and the savings from its use. [19]

This section describes the evaluation of economic efficiency while the implementation of the algorithm of spatial search.

4.2 Calculating the cost for developing the program

The cost for developing is determined by actual costs. The calculation of the estimated cost is as follows[20]:

$$S_{project} = S_{mat}+S_{spec.eq}+S_{b.wage}+S_{add.wage}+S_{soc.fond}+S_{over}+S_{add} \quad (4.1)$$

S_{mat} – is the cost of the material.
$S_{spec.eq}$ – is the cost of special equipment.
$S_{b.wage}$ – is basic salary of staff.
$S_{add.wage}$ – is additional salary of staff.
$S_{soc.fond}$ – is social contributions.
S_{over} – is overhead.
S_{add} – is additional costs.

4.2.1 Costs of the materials

Cost of materials will include such costs as paper, CDs, flash drives, folders, pens.

Table 4.1

Costs of the materials

Material	Quantity	Price per Unit. (KZT)	Amount (KZT)
Paper	1 pack	700	700
CD	1 piece	100	100
Folder	1 piece	500	500
Flash card	1 piece	3000	3000
Pen	10 pieces	100	1000
In total:			$S_{mat.}$=5300

4.2.2 Costs of special equipments

Special equipment used for carrying out this work:
Microsoft Visual Studio 2012
Visual Studio types:
1) Visual Studio Professional (OLP)
License OLP (Open License Program) includes the distribution, activation key, sticker confirming the legality of using the installed operating system.
2) Visual Studio Professional with subscription MSDN (OLP)
MSDN subscription for Visual Studio Professional provides many

advantages, in particular, it allows for developers unlimited number of any Microsoft products, as earlier, and gives excellent new versions.

3) Visual Studio Premium with subscription MSDN (OLP)

The main benefits of using Version Premium:

- Full functionality MSDN Professional.
- Automated testing interface applications.
- The controller duplicating code.
- Identification of the coverage test (code coverage analysis).

4) Visual Studio Ultimate with subscription MSDN (OLP)

The main advantages of using the version of Ultimate:

- Full functionality MSDN Premium.
- Analysis of dependencies in code using visualization.
- Visualization of the actual and potential impact of changes in the code.
- Advanced functional testing.

Best option to optimize of spatial search is second choice.

Table 4.2

Costs of special equipments

Product	**Quantity**	**Price per unit(KZT)**	**Amount(KZT)**
Visual Studio Professional	1	200000	200000
In total:			$S_{spec.eq}$ =200000

4.2.3 Basic salary of the staff

The basic salary is determined on the basis of a monthly payment of workers and labor input by the formula [20]:

$$S_{b.wage} = C_{month} * T, \quad (4.2)$$

where:

C_{month} – is monthly flat rate of employee (KZT/month);

T – is the complexity of the job.

Wages of workers to implement the project

Table 4.3

Basic salary of staff

Position of the employee	**Quantity**	**Salary (KZT)**	**Amount(KZT)**
Developer	1	130000	130000
Dispatcher	1	80000	80000
In total:			$S_{b.wage}$= 210000

4.2.4 Additional salary of staff

Additional salary is received in the amount of 20% of the basic salary [20]:

$$S_{add.wage} = 20\% \; S_{b.wage} \quad (4.3)$$

Table 4.4

Additional salary of staff

Position of the employee	Quantity	Salary (KZT)	Additional wage(KZT)
Developer	1	130000	26000
Dispatcher	1	80000	16000
In total:			$S_{add.wage}$= 42000

4.2.5 Social contributions

One must take into account the payments to the pension funds, which currently make up [21]:

- 10% - are contributions to the pension fund, which include the funded part and insurance of the pension;
- 5% - are Allocations for social tax..

Table 4.5

Social contributions

Position of the employee	Quantity	Salary (KZT)	Pension fund (KZT)	Tax (KZT)
Developer	1	130000	13000	6500
Dispatcher	1	80000	8000	4000
In total:				$S_{soc.fond}$ = 31500

4.2.6 Overhead

Overhead costs account is 80% of the basic salary fund of all primary stuff, who works on project [22].

Table 4.6

Overhead

Position of the employee	Quantity	Salary (KZT)	Overhead costs(KZT)
Developer	1	130000	104000
Dispatcher	1	80000	6400
In total:			$S_{over.}$ = 168000

4.2.7 Additional costs

Developing of project will include such costs as the cost of electricity, the cost of communication services (telephone, internet), the cost of stationery. That is shown in Table 4.7.

Table 4.7

Additional costs

Name	Price(month/KZT)
Electricity	50000
Connection	50000
In total:	S_{add} = 100000

4.2.8 Total costs

Calculation of costs and their share, the total costs are shown in the Table 4.8.

Table 4.8

Total costs

Name	Price(month/KZT)
Material	5300
Special equipment	200000
The basic wage	210000
Additional wage	42000
The Social Fund	31500
Overhead	168000
Additional expenses	100000
In total:	$S_{project}$ = 756800

4.3 The calculation of the economic efficiency of the system of spatial search.

Cost-effectiveness is based on comparison of the complexity of the various operations in the ordinary search and using the spatial search. All data and calculations are shown in the Table 4.9[23]

Table 4.9

Capital expenditures for the development

Name	Designation	Unit	Input
Average number of queries	N	Hour	1300
The time taken to process the request (organic search)	T0	Hour	0,003
The time taken to process the request	T1	Hour	0,001

Table 4.9 continuation

The average salary in Kazakhstan	S	tenge./ hour	1 000
Capital expenditures for the development	Kp	tenge.	756800

Ongoing annual savings of operating costs is calculated by the formula [24]:

$$Sa_{.} = A - D_p \quad (4.4)$$

Where:

A – is annual running costs for the calculation of the usual method of searching,

D_p – is annual running costs for the calculation of optimized search method.

The average number of days of using the test system in a year is 365 days, including all holidays.

To calculate the annual operating costs in the calculation of the usual method of searching we find the annual labor input for the base case [24]:

$$T0=365*(N*T0) \quad (4.5)$$

where , 365 is the number of working days in a year that will be used by the system;

$$T0= 365 * 130 * 0,003 = 1423,5 \text{ hour}$$

To calculate the annual operating costs in the calculation of optimized find annual labor input method for the base case:

$$T1=365 *(N*T1); \quad (4.6)$$

Then the result will be: T1= 365 * 130 * 0,001 = 474,5 hour

Before the introduction of the spatial search algorithm is much more time consuming and effort requiring to process queries and to determine the nearest location or town from where the request was sent. After the introduction of the spatial search it considerably saves energy, time of the citizens of Kazakhstan, which will take wages unchanged.

The average wage in Kazakhstan on the tariff rate of 1 000 tenge per hour.

As a result, the annual running costs for the normal search [24]:

$$A = T1 \cdot S = 1423{,}5 \cdot 1000 = 1423500 \ tenge \quad (4.6)$$

Annual operating costs for spatial search:

$$Dp = T2 \cdot S = 474{,}5 \cdot 1000 = 474500 \ tenge$$

As you can see from the above calculations, the annual running costs for the normal search is much more than a spatial search. But our task is to compare the normal modes and the spatial search. As a result, the current operating cost savings

is[24]:

$$Sa = A - Dp = 1423500 - 474500 = 949000 \text{ tenge.} \qquad (4.7)$$

Shareware annual economic efficiency is calculated by the formula [24]:

$$S = Sa - E \cdot C_{add} \qquad (4.8)$$

Where:

E - is normative coefficient of relative effectiveness of capital investments = 0,4

C_{add} - is capital costs for the development and implementation of software system.

$$S = 949000 - 0{,}4 \cdot 756800 = 646280$$

The payback period of full development costs (the length of time during which development costs can be recovered through the annual savings in operational costs) is given by[24]:

$$T_{ок.} = K_{доп} / Э_{г}; \qquad (4.9)$$

$$T_{ок.} = 756800 / 646280 = 1{,}2 \text{ years} \sim 1 \text{ year and 2,4 months.}$$

Measures of economic performance are summarized in Table 4.10

Table 4.10

Payback period

	Indicator	**Unit**	**The value of the indicator**	
			Normal search	**Spatial search**
	The costs of processing the request	Hour	0,003	0,001
	Annual labor input	hour /year	1423,5	474,5
	Annual operating costs	tg./year	1423500	474500
	Annual economic efficiency	tg./year	646280	
	Payback period	Month	1 year and 2,4 months	

4.4 Economic analysis of the project

Finally it is shown that the algorithm of spatial search provides the same quality of service as the usual search, but the cost of maintenance and use is much less than the use of usual search. Spatial search require much less time for searching.

The given calculations showed that this project about implementation of algorithm of spatial search is absolutely economically feasible. The main economic indicators of the project were:

- The payback period of the project made 1 year and 2.4 months that allows accepting the decision in favor of this project in present conditions in the market of small business enterprises;

- Calculations showed that implementation of this project will be economically expedient. In other word, this development is a cost-effective and recoverable, so it is suitable for its implementation.

5 LABOR PROTECTION AND INDUSTRIAL ECOLOGY

5.1 General information about labor protection in the company.

According to the Labor Code of 2007 on May 15, Article 24 of the Republic of Kazakhstan, healthy and safe working conditions should be set up in enterprises, institutions and organizations.

The work on the optimization of the spatial search was conducted in the Joint-Stock Company "National Information Technologies". The area of the workplace - 15m2. Dimensions of doors are 1.8 x0, 7 m. There are 6 computers indoors. Hardware parameters:

- System unit
- Power Supply 350W
- CPU Dual Core E 5300 - 2.60 GHz
- Motherboard MB Elite Group P4VXAD S-478 power 15-20 W
- Memory RAM - 2GB
- Hard Drive Seagate 40 GB 7200 power 10-15 W
- Video card GForse 2128 MB 25W, 5V ~ 5A
- Multi flash reader USB device
- Hp DVD A DH16ALL ATA Device
- Monitor 15 "LG StudioWorks 520Si 800x600-80Hz
- Power 150 watt, 100V ~ 1.5A

5.2 Analysis of dangerous and harmful factors.

When one works with a computer person is exposed to a number of dangerous and harmful factors:

- High voltage electrical current;
- Electrostatic field (ESP);
- The electromagnetic field (the range of radio frequencies: HF, UHF and SHF);
- Infrared and ionizing radiation;
- Noise and vibration.

The working conditions at the workplace are formed under the influence of a large number of factors that differ in nature, forms, form of activity. For example, electricity, lack of lighting, etc.

Actions of hazard factors have random probabilistic nature, they are difficult to predict and anticipate. Working safety requirements for equipment, tools, devices, processes and other sources of danger aimed at prevention of exposure, hazardous factors. Depending on the duration and quantity, hazards can become dangerous.

Let us consider those that are directly related to the theme of this project.

5.3 Production Sanitation and occupational health

5.3.1 Electromagnetic radiation

Electromagnetic radiation. It's impact on a person depends on the electric and magnetic fields, energy flow, the oscillation frequency, the size of the irradiated surface of the body and the individual characteristics of the organism.

Adverse biological effects manifested when the electric field is more than 20 V / m for frequencies from 60 kHz to 30 MHz and the magnetic field strength is greater than 5 A / m for frequencies from 5 kHz to 300 MHz.

The structure of the PC monitor screens has substances (strontium, lead) that the level of X-ray radiation at a distance of 5 cm from the screen did not exceed 0.03 micro roentgen / sec, so special arrangements for radiation protection are not required.

Static electricity. One of the possible hazards when working with computer facilities are electrostatic field. Static electricity – is a phenomenon caused by the accumulation and concentration of electric charges in the process of electrification. Static electricity affects people due to electrification of the interior floor at low relative humidity (below 40-45%). Voltage negative charges on the surface of floor varies from 40 - 800V, with the highest value at a relative humidity of less than 35%.

To reduce the quantities occurring when static electricity is used: coating technology of anti-static linoleum from polyvinyl chloride brand ACH, local humidification have to be considered.

Electrification also appears in the work of the CRT - monitors. First of all, electrified glass screen, some charge appearing in the whole body monitor. It is recommended to use monitors with built-in anti-static shielding against static electricity. If the selected monitor type does not provide protection from static electricity, one needs to use special protective filters, following the instructions on how to install and to ground.

5.3.2. Industrial noise.

Noise – is the mechanical vibrations in solids, liquids or gases. The impact of noise on the body depends on the volume, pitch, tone, sound and duration of exposure. Let's distinguish between permanent and non-permanent noise. During long-term exposure to noise auditory acuity is reduced, blood pressure changes, weakened attention, deteriorating vision appears, there is a change in the respiratory centers.

The sound is characterized by a frequency, intensity and pressure. Noise vibration which frequency is in the range of 20-20000 Hz, is perceived by the human ear as sound. Below 20Hz – is infrasound, above 20,000 Hz – is ultrasound, which does not cause auditory sensations, but have a biological effect on humans.

Total allowable sound pressure level is 60 dB premises.

Table 5.1

Frequency and level of noise

frequency, Hz	31,5	3	25	250	500	1000	2000	4000	8000
noise level, dB	86	71	61	54	49	45	42	40	38

Construction and acoustic methods of protection:

• Sound insulation envelope, seal around the perimeter of the chapels of windows and doors;

• sound-absorbing structures and screens;

Noise sources for software engineer in the workplace are usually hardware (computer, printer, ventilation equipment), as well as external noise. They make quite some noise, so the room is enough to use sound absorption. Reducing the noise which is penetrating into the space from outside the perimeter is achieved by seal of the windows and doors. Acoustic absorption means the property acoustically treated surfaces to reduce the intensity of the reflected waves are carried out by converting sound energy into heat energy. Sound absorption is quite effective measure to reduce the noise. The most pronounced sound-absorbing properties are fibrous and porous materials: fiberboard plate, fiberglass, mineral wool, the porous polyvinyl chloride, etc. By the sound-absorbing materials are only those that which sound absorption coefficient is not lower than 0.2. Sound-absorbing lining of these materials (for example, mats of super thin glass fiber sheathed must be placed in glass on the ceiling and upper walls). Absorption maximum is reached when wall is not less than 60% of the total area of the enclosing surfaces of the room.

For example. The noise level arising from a few incoherent sources which operating at the same time, were calculated on the basis of the principle of summation of individual energy sources [25]:

$$L_{\Sigma} = 10 \lg \sum_{i=1}^{i=n} 10^{0.1 L_i}, \qquad (5.1)$$

Where:

L_i – is sound pressure level of the i-th source;

n – is the number of noise sources.

The results are compared with the permissible noise level for a given workplace. If the calculation results are more than allowable noise level, then we need special measures to reduce the noise, by methods given previously.

The sound pressure levels of noise sources operating on the operator in the workplace are shown in the Table 5.2 [25].

Table5.2

Noise sources

Noise source	**The noise level, dB**
Hard disk	40
Fan	45

Table 5.2 continuation

Monitor	17
Keyboard	10
Printer	45
Scanner	42

Usually operator station is equipped with the following equipment: hard drive in the system unit, the cooling fan of PC, monitor, keyboard, printer and scanner.

Substituting the values of sound pressure level for each type of equipment in the formula, we get:

$$L_{\Sigma} = 10 \cdot \lg (10^{4} + 10^{4,5} + 10^{1,7} + 10^{1} + 10^{4,5} + 10^{4,2}) = 49,5 \text{ dB}$$

The resulting value does not exceed the permissible noise level for the operator position, equal to 65 dB [25]. And when one considers that it is unlikely that peripherals such as a scanner and a printer will be used at the same time, this figure would be even lower. In addition, the printer is not necessarily the immediate presence of the operator, as printer is provided with a mechanism for document feeders.

5.3.3. Lighting of work places.

Workplace lighting must be so well that the worker can do its job without voltage of view.

Fatigue of vision depends on:

- Low light conditions (leading to eye strain, fatigue and premature weakening attention);
- Too much light (causes blindness, irritation and pain in the eyes);
- Wrong direction of light (can create sharp reflections and shadows, disorient worker);
- Abrupt transitions from one brightness of the field of view to the other (causes so long - can last for minutes - the adaptation of view);
- Pulse light (cause of fatigue, decreased performance, can cause a strobe effect).

Lighting intensity of each type of work depends on:

- The size of the object;
- The contrast of the object with the background;
- Precision work.

Properly designed and executed lighting provides a high level of efficiency, has a positive psychological impact on workers to increase productivity.

Norma lighting facilities personnel performing work on the PC with the horizontal plane of light regulation and its height 0.8 m from the floor:

- With the combined light - 750 lux;
- With ambient lighting - 400 lux;

- Ripple factor - not more than 15%.

Recommended Illumination to work with display screen – is 200 lux, and when working with the screen in conjunction with the work on documentation – is 400 lux. Recommended brightness in the field of view of operators should be in the range of 1:5 - 1:10.

Calculation of lighting in the room. Requirements for lighting in the room with computers are as follows: when the visual work of high accuracy is performing, general illumination shall be E_n = 300lux and combined E_n = 750lux; similar requirements are applied when working with medium accuracy E_n = 200 and E_n = 300lux respectively.

Calculation of the area of skylights to provide natural light.

The calculation is as follows[26]:

(5.2)

Where:

S_0 – is windows area, equal to 6 м^2;

S_n – is the floor area, equal to 15 м^2;

j_0 – is characteristic of window's light, j_0=11,5 [26];

τ_0 – is total light transmittance of the light aperture:

$$\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4 \cdot \tau_5 \quad (5.3)$$

Where:

τ_1 – is coefficient of light transmission aperture, for dual flat glass, τ_1=0,8 [26];

τ_2 – is coefficient of the loss of light on the wooden aperture of window, τ_2=0,7 [26];

τ_3 – is factor taking into account the loss in load-bearing structures, τ_3=1 [26];

τ_4 – is coefficient for the loss of light in shading devices, τ_4=1 [26];

τ_5 – is factor taking into account the loss of light in the protective grille installed under the lamps τ_5=0,9 [26];

$\tau_0 = 0{,}8 \cdot 0{,}7 \cdot 1 \cdot 1 \cdot 0{,}9 = 0{,}5$

γ_1 –is coefficient reflecting the increase in the coefficient of natural light at side light, because of the light reflected from the surface of the room (walls, ceiling), γ_1=3 [26];

$k_{зд}$ - is factor considering blackout windows opposite buildings, $k_{зд}$=1,4 [26];

$k_з$ – is safety factor, $k_з$=1,5 [26];

e_n – is normalized value of the coefficient of natural light:

$$e_n = e \cdot m \cdot c, \quad (4.3)$$

e – is coefficient of natural light in % at diffused light from the sky, e=2% [26];

c – is factor sunshine climate, c=1 [26];

m – is ratio of the light climate, m=1 [26];

$$e_n=2*1*1=2\%$$

The required area of windows:

$$\left(100\cdot\frac{S_0}{S_n}\right)_{requ}=2\cdot11{,}5\cdot1{,}5\cdot\frac{1{,}4}{0{,}5\cdot3}=32\ \%$$

The actual area of windows:

$(100 \cdot S_0/S_n)_{actu} = 100 \cdot 6/15 = 40\%$

Therefore, lighting of the department meets health standards SNP 23.05.91 [26].

Calculation of artificial lighting.

The calculation is made for the rooms with area - 15 м^2, with a width - 5m, height - 3 m. Let's use the method of flux [26].

To determine the number of fixtures we define the luminous flux incident on the surface by the formula:

$$F=\frac{EKSZ}{n}\quad()$$

Where:

F – is calculated luminous flux, lux;

E – is normalized minimum illumination, lux. The work of engineer-manager may be classified as precision work, therefore, the minimum illumination will be E_n = 300 lux;

S – is the illuminated area of the room (in our case S = 15 м^2);

Z – is the ratio of average brightness to the minimum (usually assumed to be 1,1…1,2 , let Z = 1,1);

K – is factor of safety, taking into account the reduction in light output as a result of contamination of the lamp lights during operation (its value depends on the type of premises and the nature of the work carried out in it, and in our case K = 1,5);

n – is utilization rate (expressed as the ratio of luminous flux incident on the estimated surface to the total flux of all lamps and is calculated as a fraction of a unit, depending on the characteristics of the lamp, the size of the room, painting walls and ceilings, which are characterized by reflection coefficients of the walls (RS) and the ceiling (RP)), the value of the coefficients of RS and RP: PS = 40%, RP = 60%. The value of n is defined by coefficients table of use different lamps [24]. To do this, we calculate the index premises as follows:

$$I=\frac{S}{h(A+B)}$$

Where:

S – is room space, S = 15 м^2;

h – is estimated depth, h = 2.92 м;

A – is width of the room, A = 3 м;

B – is length of the room, B = 5 м.

Substituting the values we get:

Knowing the index premises I, according to table 7 [26] we find n = 0,22

We substitute all the values in the formula for determining the luminous flux:

We get: F_1 = 33750 Лм

For fluorescent lighting, we select the type LB40-1, the light output is F_2 = 4320 lux;

Calculate the number of lamps required by the formula:

$N= F_1/F_2=8$

When choosing lighting luminaries use ML. Each lamp comes with two lamps.

5.4. Electrical safety.

Passing through the human body, the electric current has a complex effect on it:

- thermal (burns, heating of tissues and biological environments);
- electrolytic (decomposition of biological fluids, changes in the physical and chemical composition);
- mechanical;
- biological.

The risk of electric shock depends on:

- Electrical Factors (voltage, current, and frequency of the current race, the resistance of the human body);
- non-electric (individual characteristics affected, duration of exposure, the current path through the body);
- environment (dust, humidity).

Technical means and methods of protection against electric shock:

- Electrical insulation of live parts with insulating materials, which limits its own resistance to the current flowing through the body by the closure.
- Protective ground – is intentional connection to ground the metal parts of the equipment that may be in the event of an accident under pressure.
- Safety shutdown – is quick protection system, automatically breaking electrical installation in the event of the risk of the person.
- Electrical separation of networks.
- Locking devices.
- PPE – is gloves, boots, mats, insulated tools.

The main methods of protection against static electricity:

- Grounding (removal of electric charges on the conductive parts of the equipment);

- The increase in the conductivity of the dielectric (increased capacity to conduct accumulating charges);

- Humidifying the air (relative humidity is 70% and more material is

accumulated in a sufficient amount of moisture to prevent accumulation of static electricity);

- Ionization of air (air saturation positive and negative ions in the field of generation and accumulation of electric charges);

- Selection of contact pairs (material selection, purchasing as a result of electrostatic discharges of different polarity);

Changing processes (elimination or reduction of friction is achieved with grease, reducing roughness, reducing the contact area of the rubbing surfaces, decreasing speed of processing, materials handling, drain the fluid, etc).

5.5. Fire safety.

The fire – is uncontrolled burning in time and space, causing property damage and endanger the lives and health of people.

The causes of fires:

- Errors in the design and manufacturing processes,

- flaws of equipment design , the wrong choice of execution of electrical equipment for hazardous areas

- The processes of spontaneous combustion

Hazards of fire – are a high ambient temperature, smoke, reduced oxygen concentration.

Fireproof object if it is excluded the possibility of fire and in the case of preventing possible human exposure and ensure protection of property in fire hazards.

Fire hazards which can affect humans lives:

- The flame and sparks;
- Increased ambient temperature;
- Toxic products of combustion;
- The smoke;
- Reduced oxygen concentration;
- Collapse and damage of buildings and structures;
- Shock Wave.

During a fire, the person is exposed to many factors:

- Burns of varying complexity;
- Suffocation from smoke until death;
- Injuries and fractures in the collapse of buildings.

The sources of fire in the laboratory may be a PC and other electrical devices which as a result of faults generate superheated elements arcs and spark which may cause fire flammable materials in the laboratory which are desks, chairs, doors burn, etc., for explosion such premises can be classified as - fire hazard.

Fire safety requirements:

- Never use electric heaters with open heating element, an open fire in the workplace. Use of electric heaters with closed heating elements is permitted only in designated areas.

- Do not overload the circuit powering with the additional equipment, household and particularly heating devices. Prohibits the unauthorized switching of computer and peripheral equipment, to avoid overheating of the individual lead wires.
- Do not put any connecting cables so that they cross passages for people.

For early detection and extinguishing fires, workplace must be equipped with:

- Automatic fire alarm system installed in a room with heat sensors;
- Evacuation plan, direction;
- hand-held carbon dioxide fire extinguisher to extinguish the electrical equipment.

The workplace does not have environmental hazards.

CONCLUSION

Science and technology are developing without a breach of continuity. In the past it was required to spend hours in order to calculate some simple expressions, today, however, we can do it in couple of seconds. And stricter requirements come with better technology. Therefore, optimisation of different algorithms will always be demanded.

Model of spatial search is developed in this paper. Points with coordinates (x,y) are represented as tree nodes. The goal of this program is localisation of this point or, in other words, determination of approximate location of coordinate. All notations used in determination of search trees are systemised in this paper. Theory of binary search trees is considered in more details. During the work principles of object-oriented programming and class templates creation were studied.

In the economic part of our thesis results of calculating the financial efficiency of the project in the implementation of investments show that the project is economically expedient. Accordingly, implementation of this project can be considered advantageous, therefore, in the future the expansion and implementation of the program is planned. Civil Services such as police, ambulance and fire department may be interested in this program.

Thus, we can say that the tasks are solved, and the purpose of the course work is achieved.

REFERENCES

1. Dyusembaeyev A.E. Mathematical models of segmentation programs. Moscow: Fizmatlit (MAIC, NAUKA), Series “Programmers library”, 2001, 208pp.

2. Gasanov E.E., Kudryavtsev V.B. Theory of information storage and retrieval. Moscow: Fizmatlit (MAIC, NAUKA), 2002.

3. Lavrov S.S. Introduction into Programming. M: Nauka, 1974.

4. Lavrov S.S. Programming. Mathematical basis, tools, theory. St. Petersburg: Publisher BHV, a series of “Master”, 2001, 318.

5. Lengsam J., Ogenstain M., Tenenbaum M. Data structures for personal computers. M: Mir, with 1989, 567.

6. Peter Andras. Kernal-Kohenen Networks. Int. Jour.of Neural systems. Vol.12, No.2,2002, p.101-119

7. Romanovsky I.V. Discrete analysis. St.P, Izd. Nevsky dialect, 2000, 240 p

8. Tanenbaum E. Computers Architecture. M: Izd. PITER, 2003, 698p.

9. Virt N. Algorithms + Data structures = program. M: Mir, 1986

10. Ferrari D. Performance evaluation of computer systems. Moscow: Mir, 1981, 585p.

11. Yablonsky S.V. Introduction to discrete mathematics. M: Nauka, Fizmatlit, ed. 3, 2000

12. Drozdek A. Data structures. Algorithm in Java. Tompson Learning., 2001

13. A.P. Ershov. Introduction to Theoretical Programming. M: Science, 1981

14. Adelson-Velsky G.M., Landis, E.M. An algorithm for information organization. DAN SSSR, t.146, #2, 1962, s.263-266

15. Aho A., Hopscroft J. Ulman, Data Structures and algoritmy. – M: ed. “Williams”, 2000

16. Korolev L.N., Mikov A.I. Informatika. Inftroduction into computer science. M: High School Publishing House, 2003, p 342.

17. Korolev L.N. Typical program of curriculum programming disciplines for bachelor’s degree in Applied Mathematics and Computer Science, “Journal of Programming.” #1, 1996. P.66-74.

18. Lavrov S.S. Introduction into Programming. M: Nauka, 1974.

19. Romanovsky I.V. Discrete analysis. St.P, Izd. Nevsky dialect, 2000, 240 p

20. Robert Alan Hill. Strategic Financial Management/ 1st edition, 2008- p.103

21. SR 2-12-77: Noise protection, 1977.

22. SR 2.2.2/2/4.1340-03. Hygienic requirements for video display terminals, personal electronic computers and the organization of work, 1996.

23. GOST 12.1.005-88. The air of working area. General hygiene requirements. M. :Izdatelstvo standartov, 1988, 14p.

24. SR 3-05-91. Natural and artificial lighting. – M.: 1991

25. GOST 12.1.019-79. Electrical safety. General requirements and range of species protection. M.: Izdatelstvo standartov, 1983, 18p.

26. Regulations for electrical installation. M.: Atomizdat, 1986.

APPENDIX

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private Graphics space;
    private Bitmap spaceBitmap = new Bitmap(1, 1);
    private Graphics graphDrawingArea;
    private Pen greenPen = new Pen(Color.Green, 1);

    int z = 0;
    Node[] nodes = new Node[10];
    Node temp = null;
    int LBD = 0;
    int RBD = 1000;
    int UBD = 500;
    int BBD = 0;
    Node rf_root = null;
    Coordinates search = new Coordinates(new Random().Next(1, 1000), new Random().Next(1, 500));



    private void button1_Click(object sender, EventArgs e)
    {

        LBD = 0;
        RBD = 1000;
        UBD = 500;
        BBD = 0;
        space = this.panel1.CreateGraphics();
        graphDrawingArea = Graphics.FromImage(spaceBitmap);
        graphDrawingArea.Clear(Color.White);

        Coordinates[] a = new Coordinates[10];
        space.FillEllipse(new SolidBrush(Color.Red), search.x , search.y , 3, 3);
        space.DrawEllipse(new Pen(Color.Red), search.x-2, search.y -2, 6, 6);

        for (int j = 0; j < 4; j++)
        {
            for (int i = 0; i < 10; i++)
            {
                a[i] = new Coordinates(new Random(DateTime.Now.Millisecond + i).Next(LBD, RBD), new Random(DateTime.Now.Millisecond + i).Next(BBD, UBD));
            }
            System.Threading.Thread.Sleep(1000);
            two_d_tree(a, 10, LBD, BBD, RBD, UBD);
        }

    }
    private void draw(Node[]a, int x0, int x1, int y0, int y1)
    {

    }
    private void bounds(Coordinates b,Node[] a)
    {
        z = 1;
        for (int i = 0; i < a.Count(); i++)
        {
            if (a[i].a.z(z) > b.z(z))
            {
                if (z == 0)
                {
                    UBD = a[i].a.z(z);
                }
                else
                    RBD = a[i].a.z(z);

                i = a[i].left_des - 1;
            }
```

```
        else
        {
            if (z == 0)
            {
                BBD = a[i].a.z(z);
            }
            else
                LBD = a[i].a.z(z);

            i = a[i].right_des - 1;
        }

         z = 1 - z;
         if (i < 0)
         {
             break;
         }

    }
    space.DrawRectangle(new Pen(Color.Red,1), LBD, BBD, RBD-LBD, UBD-BBD);
}
private int ToInt16(string p)
{
    throw new NotImplementedException();
}

private class Coordinates
{
    public int x;
    public int y;

    public Coordinates(int x, int y){
        this.x = x;
        this.y = y;
    }
    public int z(int z)
    {
        if(z == 0){
            return y;
        }
        else{
            return x;
        }
    }
}

private class Node
{
    public int left_des;
    public int right_des;
    public int anc;
    public   Coordinates a;

    public Node(Coordinates a){
        this.a = a;
        anc = -1;
    }

}
private Node tree(int x,int y, int i, int j)
{

    int zBound1;
    int zBound2;
    Coordinates tempp = new Coordinates(x, y);
    Node temp1 = new Node(tempp);

    if (j == -1)
    {
        rf_root = temp1;
        space.DrawLine(greenPen, x, BBD, x, UBD);
    }
    else
    {
        if (nodes[j] != null && j>=0)
        {
            temp = nodes[j];

            if (tempp.z(z) <= nodes[j].a.z(z))
            {
                zBound1 = nodes[j].a.z(z);
                nodes[j].left_des = i;
            }
            else
            {
                zBound2 = nodes[j].a.z(z);
                nodes[j].right_des = i;
            }

            if (z == 0)
            {
                zBound1 = UBD;
                zBound2 = BBD;
                space.DrawLine(greenPen, x, zBound1, x, zBound2);
            }
            else
            {
                zBound1 = RBD;
                zBound2 = LBD;
                space.DrawLine(greenPen, zBound2, y, zBound1, y);
            }
        }
    }

    //v=temp1;
    temp1.left_des = -1;
    temp1.right_des = -1;
    temp1.anc = j;

    return temp1;
}
```

```
        private void two_d_tree(Coordinates[] a, int n, int x0, int y0, int x1, int y1)
        {
            space.DrawRectangle(greenPen, x0, y0, x1 - x0, y1 - y0);
            int rf_root_i = -1;
            for (int i = 0; i < n; i++)
            {
                LBD = x0;
                RBD = x1;
                UBD = y1;
                BBD = y0;

                z = 0;
                for (int j = 0; j < i; j++)
                {
                    rf_root_i = j;
                    z = 1 - z;
                    if (a[i].z(z) > nodes[j].a.z(z))
                    {
                        if (z == 1)
                        {
                            LBD = nodes[j].a.z(z);
                        }
                        else
                            BBD = nodes[j].a.z(z);
                        if (nodes[j].right_des < 0)
                            break;
                        else
                            j = nodes[j].right_des - 1;

                    }
                    else
                    {
                        if (z == 1)
                        {
                            RBD = nodes[j].a.z(z);
                        }
                        else
                        {
                            UBD = nodes[j].a.z(z);
                        }
                        if (nodes[j].left_des < 0)
                            break;
                        else
                            j = nodes[j].left_des - 1;
                    }
                }
                nodes[i] = tree(a[i].x, a[i].y, i, rf_root_i);
            }

            LBD = x0;
            RBD = x1;
            UBD = y1;
            BBD = y0;
            bounds(search, nodes);
        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }

        private void panel1_Paint_1(object sender, PaintEventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

    }
}
```

Listing A