Table of Contents

Chapter 1

Introduction

Backtracking:

This is a recursive algorithm strategy called *backtracking*. A backtracking algorithm tries to build a solution to a computational problem incrementally. Whenever the algorithm needs to decide between two alternatives to the next component of the solution, it simply tries both options recursively.

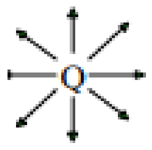
Backtracking is kind of solving a problem by trial and error. However, it is a well organized trial and error. We make sure that we never try the same thing twice. We also make sure that if the problem is finite we will eventually try all possibilities (assuming there is enough computing power to try all possibilities).

The n-Queens problem:

The prototypical backtracking problem is the classical ***n-Queens Problem***, first proposed by German chess enthusiast Max Bezzel in 1848 for the standard 8_8 board, and both solved and generalized to larger boards by Franz Nauck in 1850. The problem is to place *n*queens on an *n*_ *n*chessboard, so that no two queens can attack each other. For readers not familiar with the rules of chess, this means that no two queens are in the same row, column, or diagonal.

NOTES:

- A queen can attack horizontally, vertically, and on both diagonals, so it is pretty hard to place several queens on one board so that they don’t attack each other.



- 2 queens or 3 queen’s problem is not solvable.

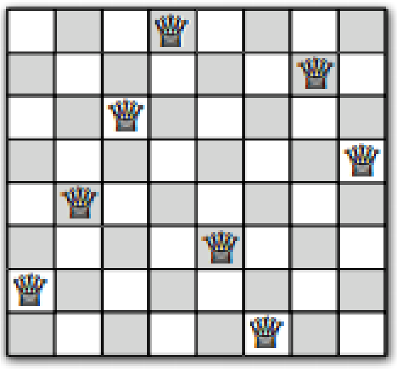
1.2 Proposed System:

- Start with one queen in the first column, first row.
- Start with another queen in the second column, first row.
- Go down with the second queen until you reach a permissible situation.
- Advance to the next column, first row, and do the same thing.
- If you cannot find a permissible situation in one column and reach the bottom of it, then we have to go back to the previous column and move one position down there. (This is the backtracking step.)

- If we reach a permissible situation in the last column of the board, then the problem is solved.
- If we have to backtrack BEFORE the first column, then the problem is not solvable.

The 8 queen problem:

Given is a chess board. A chess board has 8x8 fields. Is it possible to place 8 queens on this board, so that no two queens can attack each other?



One solution to the 8 queens problem, represented by the array [4,7,3,8,2,5,1,6]

1.3 Scope:

In chess, a queen can move as far as she pleases, horizontally, vertically, or diagonally. A chess board has N rows and N columns. The standard N by N Queen's problem asks how to place N queens on an ordinary chess board so that none of them can hit any other in one move. Besides being an amusing puzzle this problem is interesting because kids love it and it's a great teaching tool in the upper grades of Elementary School.

The n-Queen problem is basically a generalized form of 8-Queen problem. In 8-Queen problem, the goal is to place 8 queens such that no queen can kill the other using standard chess queen moves. So, in this paper, the proposed solution will be applied to 8-Queen problem. The solution can very easily be extended to the generalized form of the problem for large values of `n'. The paper contains the detail discussion of problem background, problem complexity, ant colony optimization (swarm intelligence) and a fair amount of experimental graphs.

Chapter 2

Software Requirement &Specification

2.1 Hardware Requirements

Processor	:	Intel Pentium 1V or more
Ram	:	512 MB or more
Cache	:	512 KB

Hard disk	:	16GB hard disk recommended for Primary partition.
Graphics display card	:	VGA or SVGA

2.2Software Requirements

Operating system	:	window XP or later
Front end software	:	Turbo C
User interface	:	CGI

2.3 System Design

2.3.1 Backtracking Algorithm

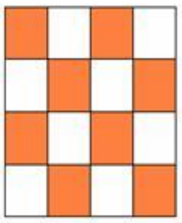
The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes then we backtrack and return false.

```
1) Start in the leftmost column
2) If all queens are placed
return true
3) Try all rows in the current column.  Do following for every tried row.
    a) If the queen can be placed safely in this row then mark this [row,
column] as part of the solution and recursively check if placing
queen here leads to a solution.
    b) If placing queen in [row, column] leads to a solution then return
true.
    c) If placing queen doesn't lead to a solution then umark this [row,
column] (Backtrack) and go to step (a) to try other rows.
3) If all rows have been tried and nothing worked, return false to trigger backtracking.
```

Consider the N Queens problem –

Discover if and how N Queens can be placed on an N x N chessboard in non-attacking position. (There is a solution to the N Queens problem for all boards of N ≥ 4)

Assume N = 4. We want to place 4 queens on the chessboard below in non-attacking position.



We will develop a backtracking algorithm to try to solve this problem. Backtracking is used to solve problems for which a sequence of objects is to be selected from a setsuch that thesequence satisfies some constraint. In the case of the N-queens, the constraint is that all N queens be placed in non-attacking position.

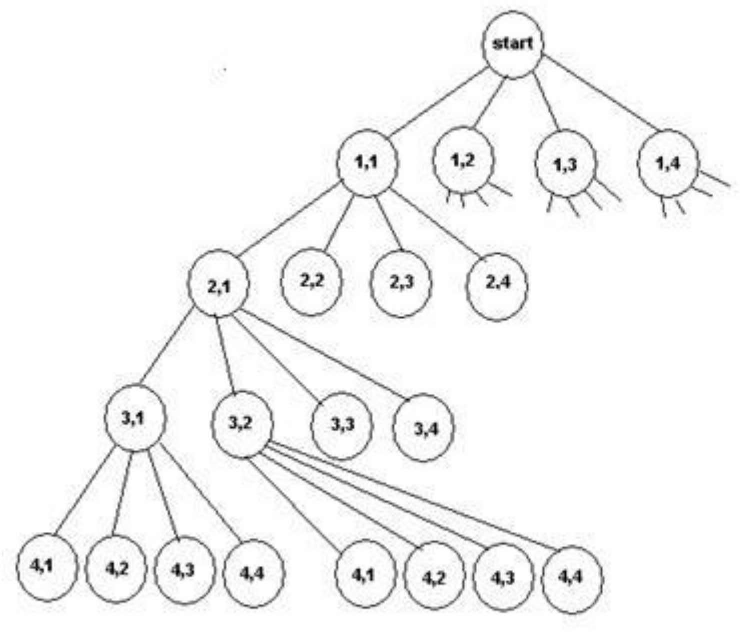
Backtracking is a modified depth-first search of a tree consisting of all of the possible sequences that can be constructed from the problem set. In backtracking we perform a depth-first traversal of this tree until we reach a node that is non-viable or non-promising, at which point we "prune the subtree" rooted at this node, and continue the depth-first traversal of the tree.

```
        public static void DFS (Graph g, Object vertex) {
```

```
    g.visit(vertex);
    Iterator itr = g.neighbors(vertex);
    while (itr.hasNext( ) ) {
        Object v = itr.next( );
        if (!g.isVisited(v) )
            DFS(g, v);
    }
}
```

This algorithm just specifies the order in which the nodes are to be "visited", it does not include any instruction for what to do at a node when it is "visited".

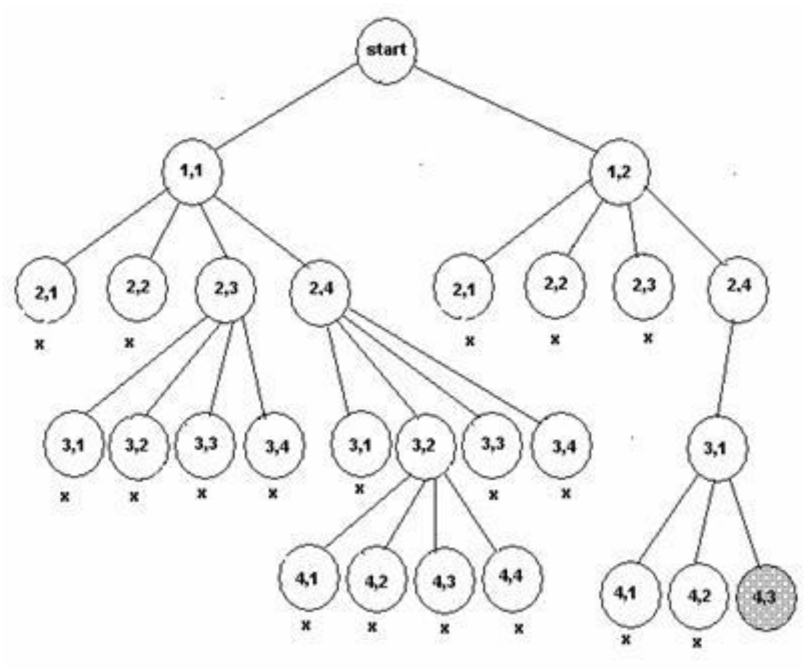
The diagram below partially illustrates the state space tree for the instance of the N-Queens problem when N = 4. the ordered pair (i, j) in each node indicates a possible row, column placement of a queen. Each queen can be placed in one of the N columns for each of the N rows, for a total of 4 x 4 x 4 x 4 = 256 candidate solutions. In general therefore, there are N^N candidate solutions.



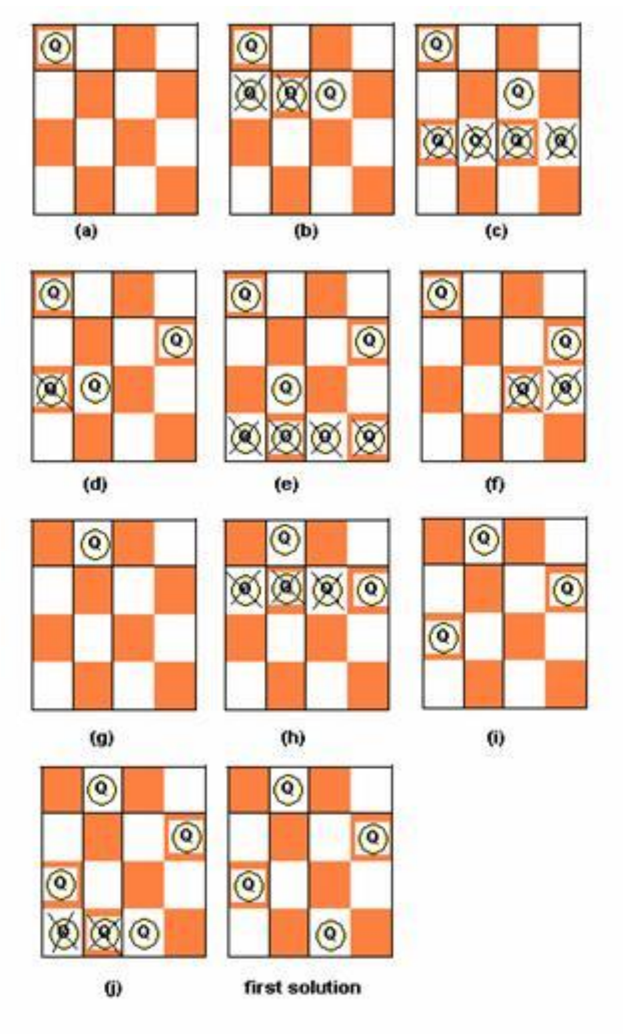
The general backtracking algorithm problem modifies depth-first search as follows:

```
public static void backtrack( node v) {
     //backtracking pseudo-code
     if (promising(v) )
          if (there is a solution at v)
               write the solution;
          else {
                for (each child u of v)
                backtrack(u);
}
```

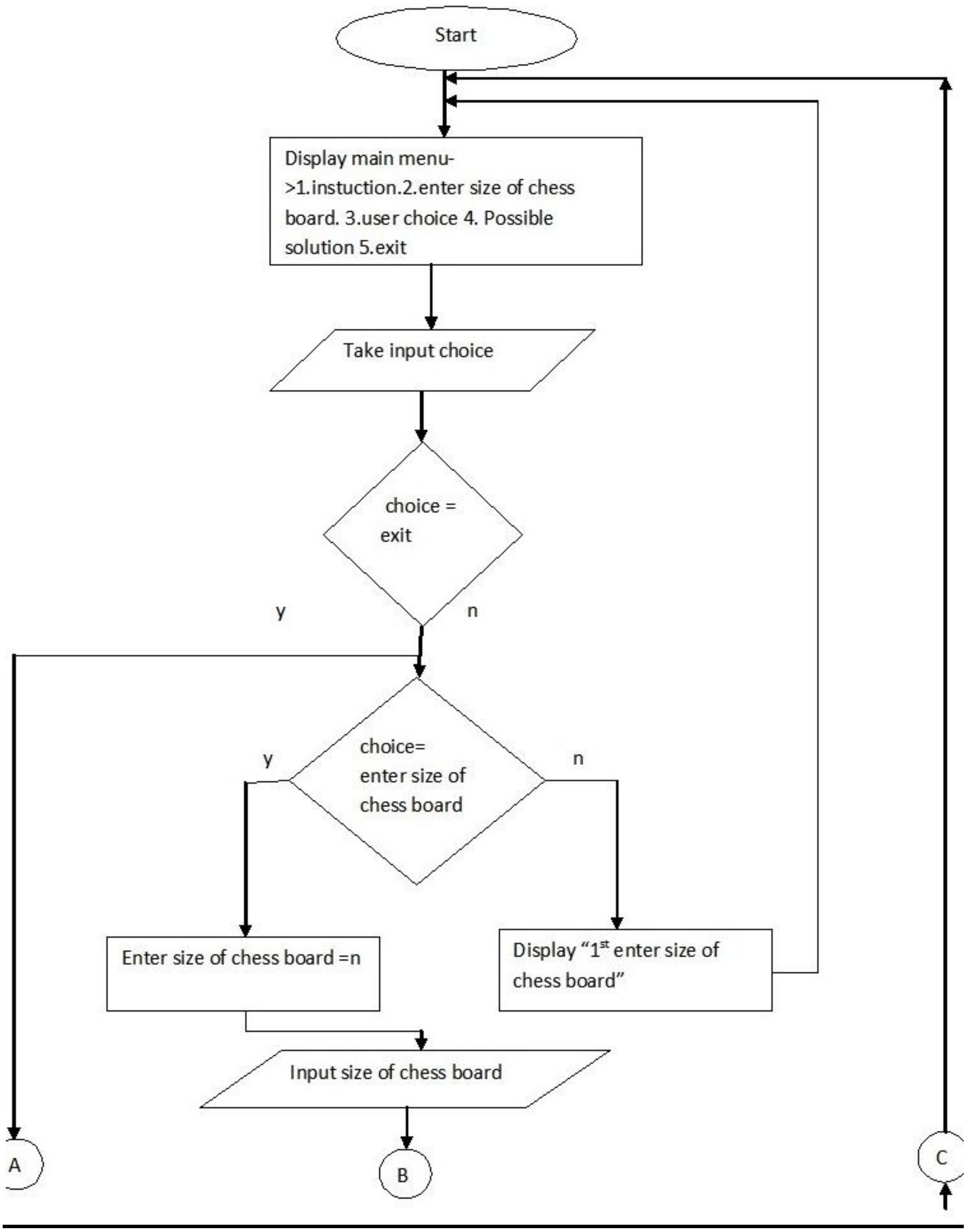
For the N-Queens problem, a placement is "promising" if the placed queen is not in the same column or same diagonal as one of the already positioned queens. To locate the first solution to the 4-queens problem, the following nodes in state-space will be visited.

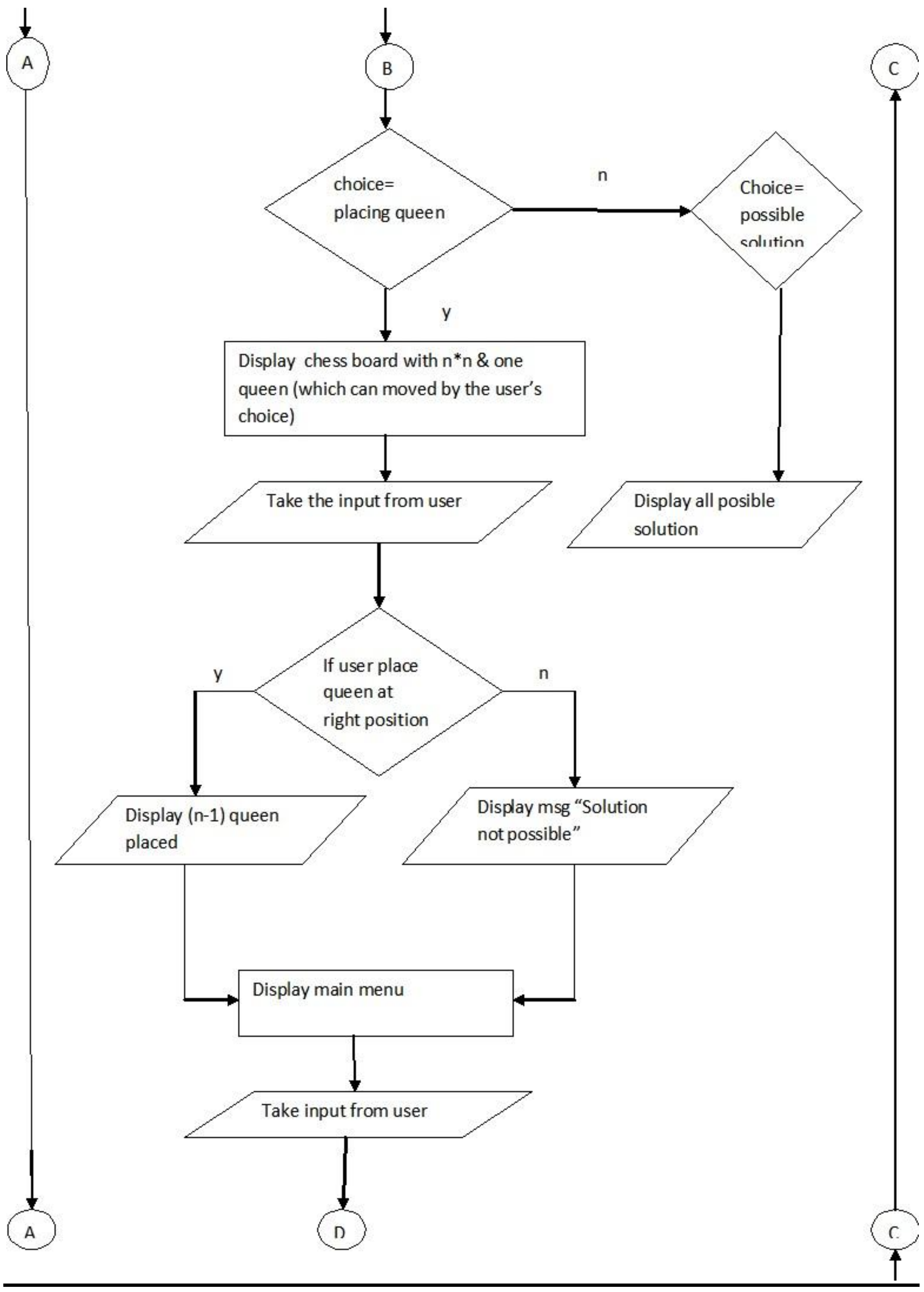


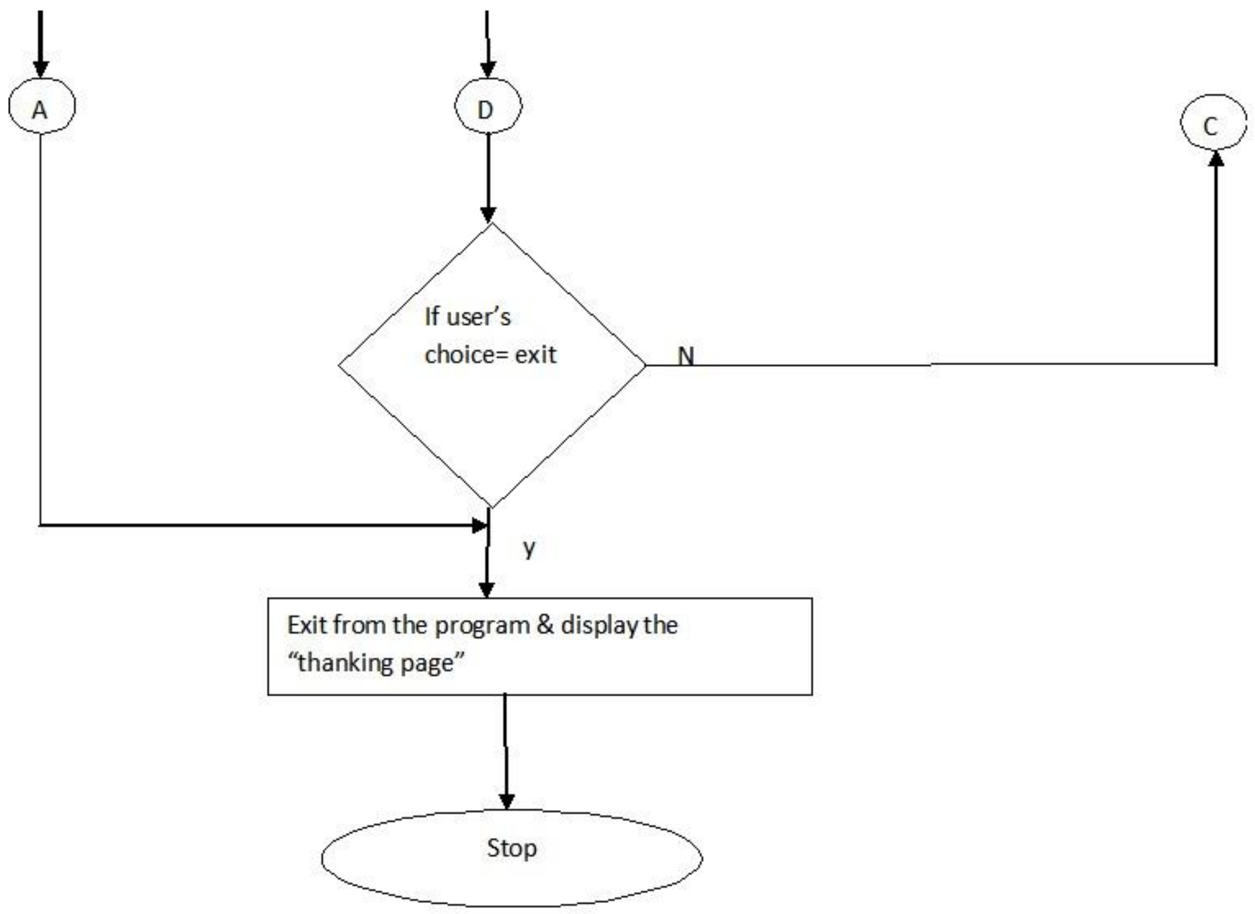
This traversal of state-space corresponds to the following attempts at positioning the queens.


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
first solution

2.4. Flowchart:


Start
Display main menu->1.instuction.2.enter size of chess board. 3.user choice 4. Possible solution 5.exit
Take input choice
choice = exit
y
n
choice= enter size of chess board
y
n
Enter size of chess board =n
Display "1st enter size of chess board"
Input size of chess board
A
B
C


A
B
C
choice=
placing queen
n
Choice=
possible
solution
y
Display chess board with n*n & one queen (which can moved by the user's choice)
Take the input from user
Display all posible solution
If user place queen at right position
y
n
Display (n-1) queen placed
Display msg "Solution not possible"
Display main menu
Take input from user
A
D
C


A
D
C
If user's choice= exit
N
y
Exit from the program & display the "thanking page"
Stop

Chapter 3

Implementation

3.1 Module Description:

Instruction:

It tells to the user about the instructions, i.e., how the user can place the queen and how the user can move the queen from one cell to a particular cell.

Enter the size of chess board

In this module, a simple message "Enter the size of chess board :"(say n) is displayed to take the input about the size of chess board from the user. The user has to input only numeric value. The number should be 4 to 10.

User choice to place queen

In this module, the (n*n) chess board is displayed on the screen with one queen at the left hand side on the top of the chess board. This one queen can be moved according to the user's choice with the help of up, down, left, right arrow key. When user wants to place the queen at a particular position then he has to move the queen on that particular position & click the enter button to place the queen at that position.

After that if user places the queen at the wrong position, a message "solution is not possible " will displayed on the screen, if not the then all (n-1) queens are placed on the chess board.

Possible solution

In this module, all the possible solutions are displayed to the user.

Exit

When user wants to exit from the application then he can exit from this module, when user click on this module then a thanking page is displayed on the screen.

3.2. Pseudo code of the main module

```
1. Start
2. Display the main menu
       Instruction
       Enter the size of chess board
       User choice to place the queen
```

```
          Possible solution
          Exit
3.  Take input from the user "choice"
         If choice == Instruction
   Display "instruction"
        Else if choice== enter size of chess board
              Flag=1; & take input size of chess board "n" from user
        Else
              Flag=0
4. If (flag==1 && choice = user choice to place the queen)
        Display the chess board with (n*n) matrix with 1 queen
          (which is moved by the user choice with the help of up,down , left,
right arrow key) & take input from user.
     If user place queen at right position then
display (n-1) queen placed according to user choice
else
display "solution is not possible"
5.  If (flag==1 && choice =possible solution)
display all unique possible solution
6.  If choice!=exit
then go to step 2
    Else
exit from the program & display thanking page
```

Chapter 4

Testing

4.1. Introduction & need of Testing:

Testing is usually relied upon to detect the faults that occur during any phase of the software development cycle, in addition to the faults that introduced during the coding phase itself. For this, different levels of testing are used which perform different tasks and aim to test different aspects of the system. The basic levels of testing are unit testing, integration testing, system and acceptance testing' the different levels of testing attempt to detect different types of faults.

Client Needs	Acceptance Testing
Requirements	System Testing
Design	Integration Test
Code	Unit Testing

4.2. Unit Testing:

This involves the tests carried out on modules programs, which make up a system. This is also known as Program Testing. The units in a system are the modules and routines that are assembled and integrated to perform a specific function. In a large system, many modules at different levels are needed. Unit testing focuses on the modules, independently of one another, to locate errors. The program should be tested for correctness of logic applied and should detect errors in coding.

For example in the OBSE system, feeding the system with all combinations of data should test all the calculations. Valid and invalid data should be created and the programs should be made to process the data to catch errors. Example for valid and invalid data check is that, in case three digit no is entered during the entry of transaction, and that no does not exists in the master file or the no entered is an exit case then the program should not allow the entry of such cases. All dates that are entered should be validated. No program should accept invalid dates. To check that are needed to be incorporated are in the month of February the date cannot be more than 28 in the case of ordinary year and 29 in the case of leap year. Also validation must be done for the months having 30 days so that only dates up to 31 can be entered. All conditions in the program must be checked and tested accordingly. Before proceeding one must make sure all the modules are working independently and safely.

4.3. System Testing:

When unit testing are satisfactorily concluded, the system, as a complete entity must be tested. At this stage, end users and operators became actively involved in testing. While testing one should also test to find discrepancies between the system and its original objective, current specifications and system documentation. System testing also verifies that the file sizes are adequate and their indexes have been built properly.

Output Testing: After performing the validation testing, the next step is output testing of the proposed system. Since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the output generated or displayed by the system under consideration. Hence the output

Module Name	Test Cases	Expected Output	Output	Result

format is considered in two ways – one is on the screen and the other on the printed format.

Testing involves exercising the program using data like the real data processed by the program. The existence of program defects or inadequacies is inferred from unexpected system outputs. For verification and validation we make use of the program testing technique.

Validation Testing:

Validation testing is the step where requirements established as a part of the software requirements analysis, are validated against the software already been developed. This test provides the final assurance that the software meets all the functionality, behavioral and performance requirement and rectifies the errors, which are uncovered during the coding.

Deviation or errors discovered at this step as corrected prior to the completion of this project with help of user negotiating to establish a method for resolving deficiencies. Thus, the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

4.4. Test cases:

Menu	a)The user chooses the option of placing the queen without selecting the size of the chessboard b)The user chooses the option: possible solutions without selecting the size of the chess board	a)Error message will be displayed to enter the size of chessboard . b) Error message will be displayed to enter the size of chessboard.	An Error message will be displayed to enter the size of chessboard. An Error message will be displayed to enter the size of chessboard.	Menu page
User choice to place the queen	a)User places the queen at wrong position on the chessboard	a)Error message: Solution not possible will be displayed	Error message: Solution not possible will be displayed	Menu Page

Chapter 5

Conclusion & Future Enhancement

5.1. Conclusion:

An N Queens backtracking algorithm is much more efficient by any brute force approach. The idea is to place one queen on one edge and then continue by placing the next queen on the first valid position (in the next row / column) and so on. When no more queens can be placed the algorithm has either found a solution (if all queens are placed) or it needs to remove the processed queen and move the previous one to the next valid position. When a queen has been placed on the last valid position in a row / column and needs to be replaced it must be removed and the previous one must be moved to the next valid position.

This project has been implemented using C graphics and the entire process of backtracking can be better understood by the user in the graphical form. Through this project, the user can also view the possible solutions to the problem since there may be more than one solution for a given condition.

5.2. Future Enhancement:

The chessboard size implemented in our project is limited to 10 which can be increased further. The user can place only one queen according to his choice whereas in future, in this project we can allow the user to place all the specified number of queens according to the chessboard size and allow him to get the solution by himself. There can also be a facility where the user can be informed whether his choice of placing the queen on the chessboard is at the correct position or not. The time complexity and space complexity can also be calculated and displayed to the user. The appearance of the Queen can also be made more attractive graphically.

Bibliography

The following were referred during the analysis and execution phase of the project,

Reference Books:

- **KanetkarYeswant.** Let us C. 5th edition, BPB publications,2005
- **Silberschatz Abraham (bell labs) & Galvin Peter Baer (Corporate technologies, inc.),** operating system concepts, 5th edition.

Web references:

- http://www.durangobill.com/N_Queens.html
- http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/
- http://stackoverflow.com/questions/7730360/all-possible-solution-of-the-n-queens-algorithm
- http://www.academic.marist.edu/~jzbv/algorithms/Backtracking.html

Chapter 7

Screen Shots

Home Page:

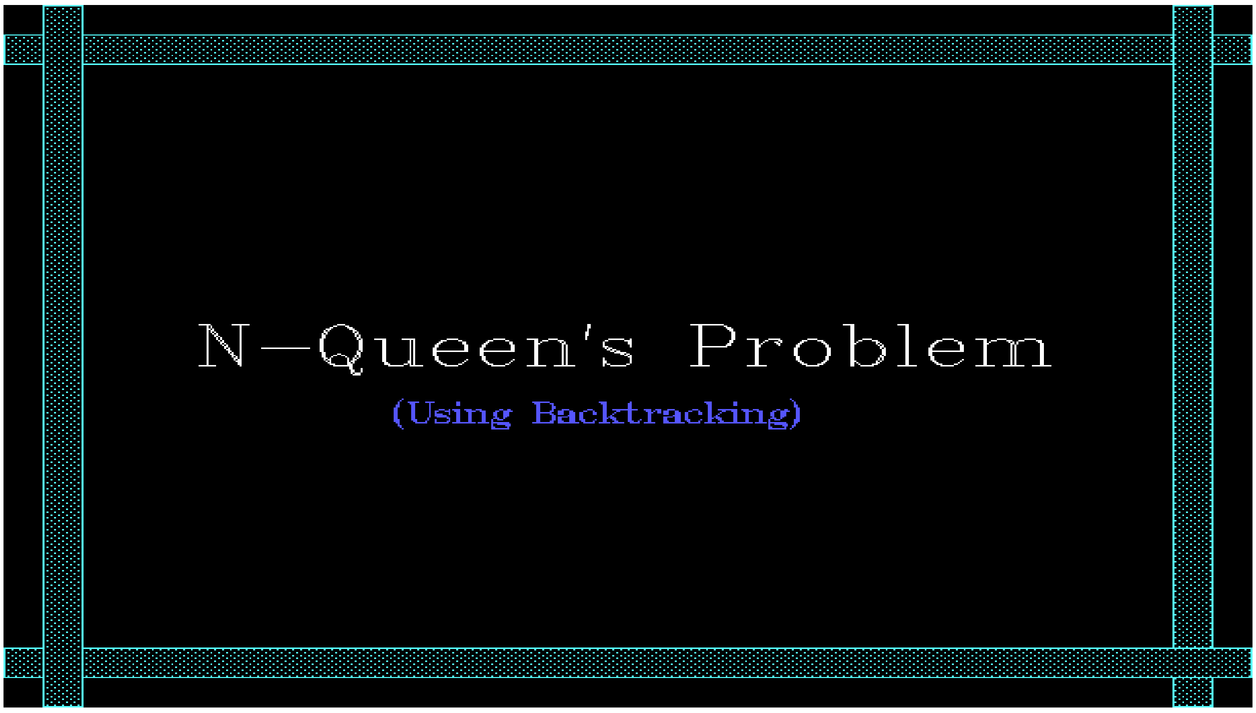


Fig 1: Home Page

Menu Page:

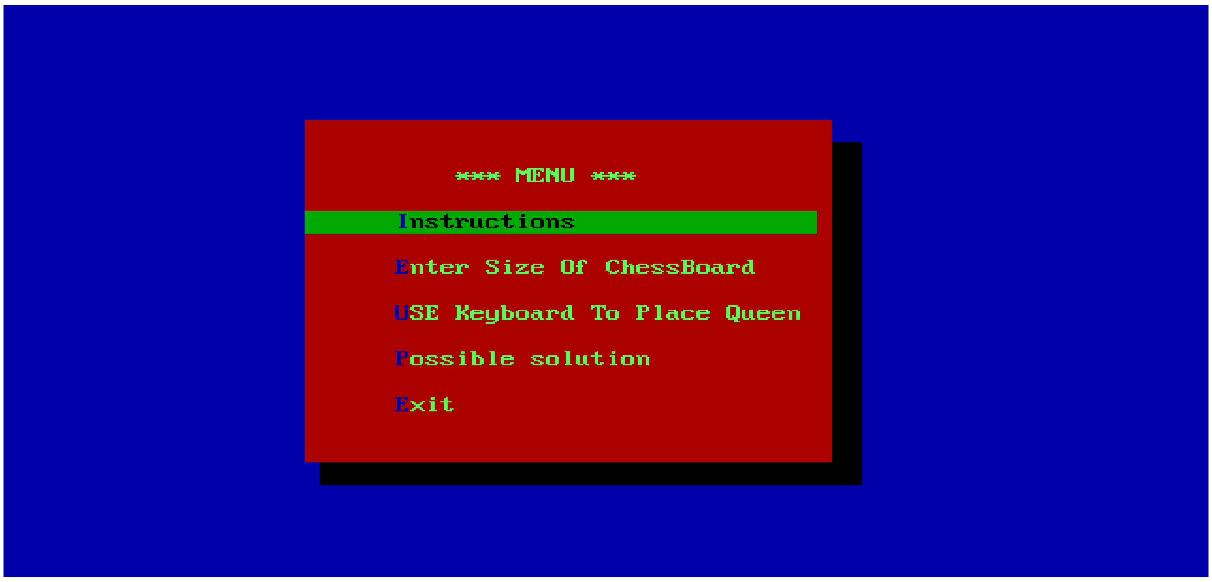
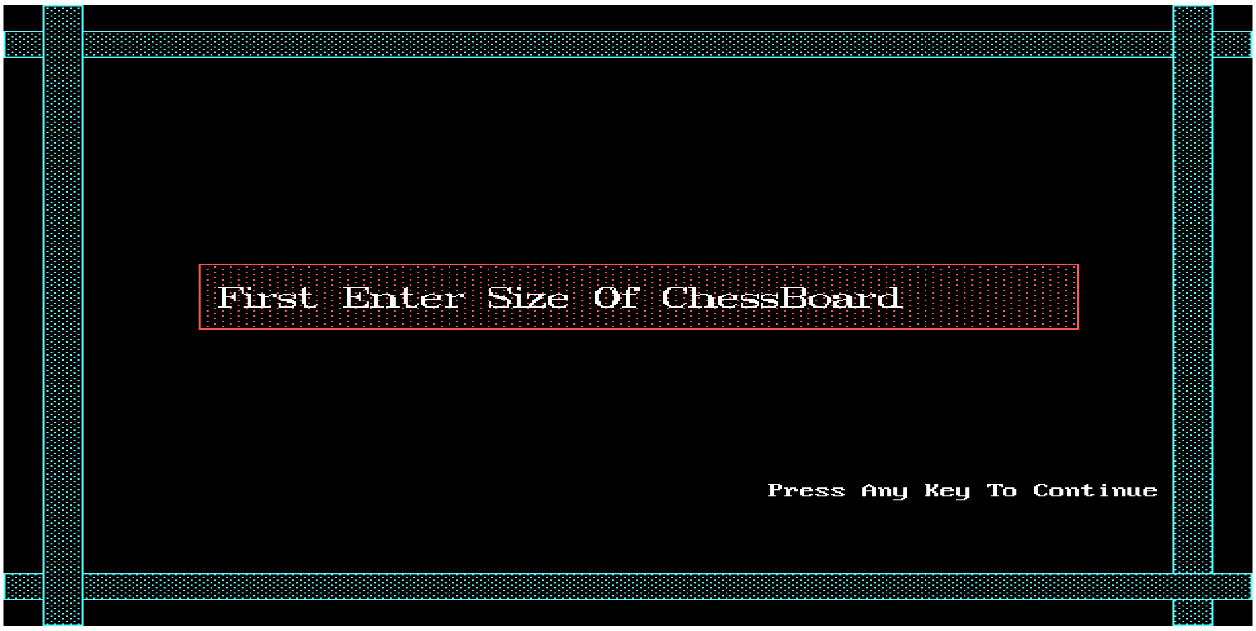


Fig 2: Menu Page

Input Page:

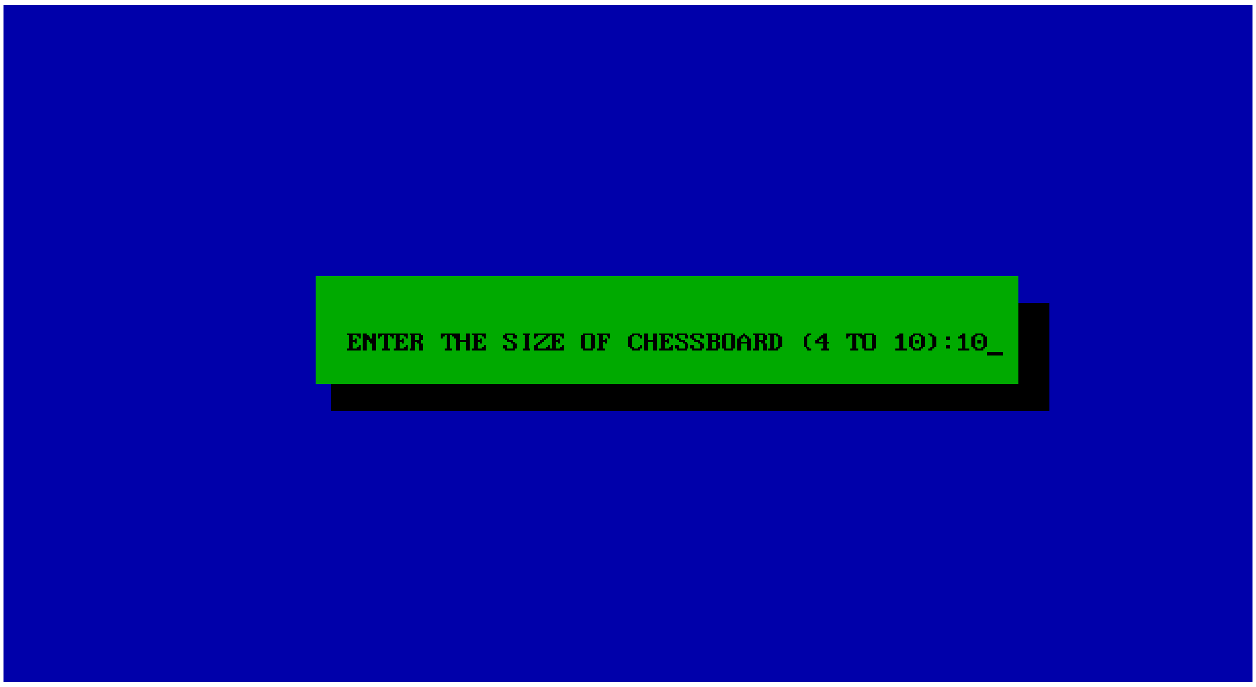


```
First Enter Size Of ChessBoard

                                   Press Any Key To Continue
```

Fig 3: Input Page:

Input Page:



```
ENTER THE SIZE OF CHESSBOARD (4 TO 10):10_
```

Fig 4: Input Page:

Output Page:

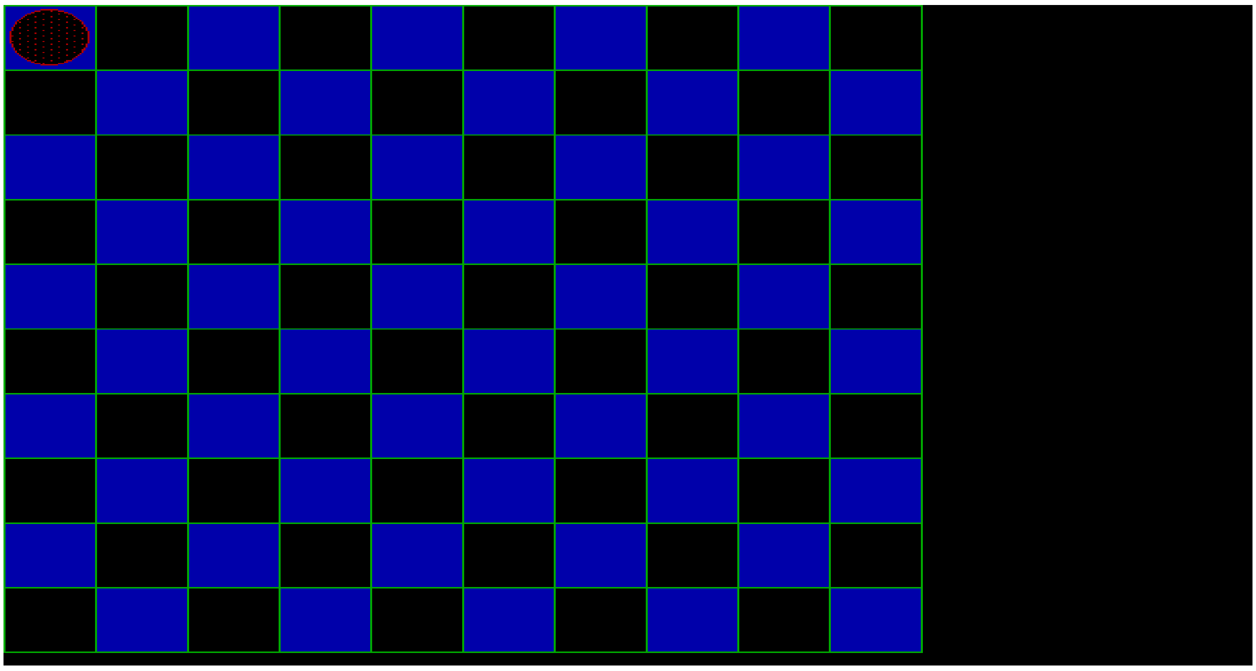
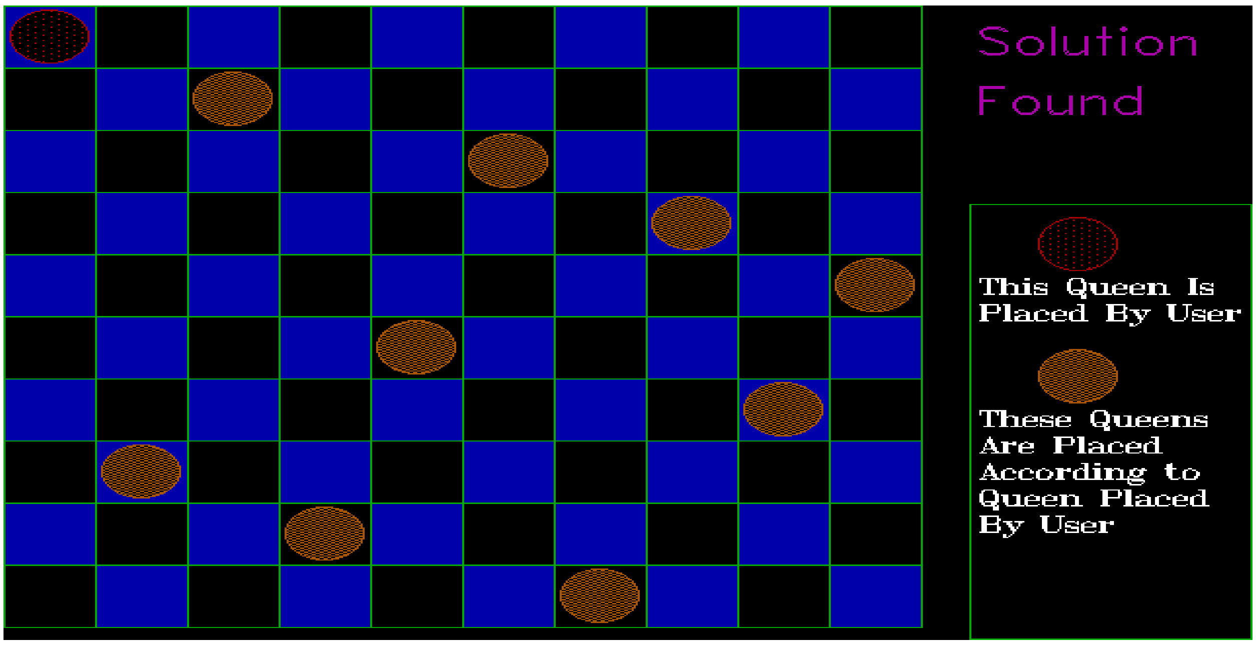


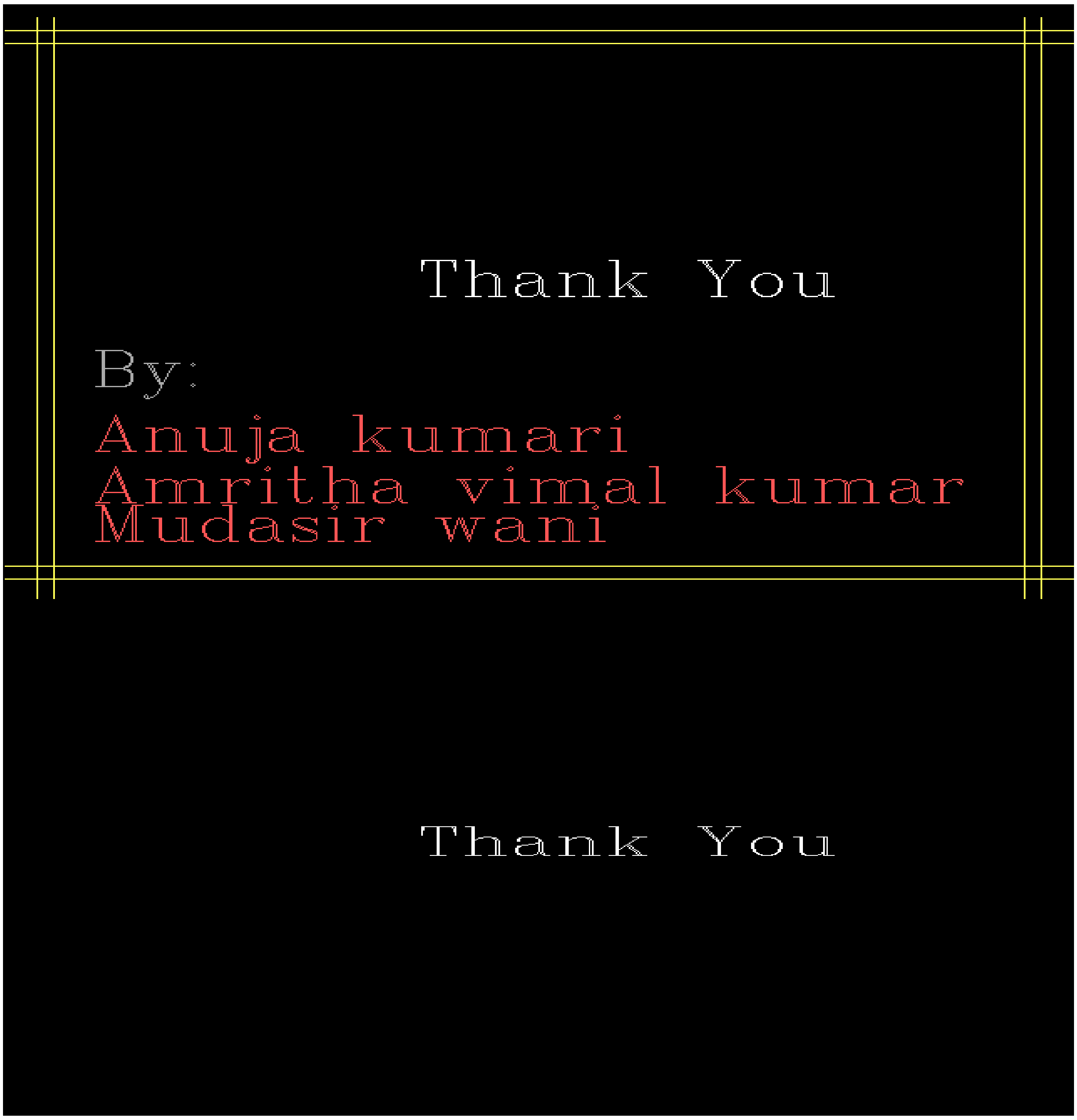
Fig 5: Output Page

Output Page:

Fig 6: Output Page:


Solution Found
This Queen Is Placed By User
These Queens Are Placed According to Queen Placed By User

Exit Page:



```
Thank You

By:
Anuja kumari
Amritha vimal kumar
Mudasir wani



Thank You
```

Fig 7: Exit Page:

Appendix

Introduction of Programming Language

Introduction of C:

Every full C program begins inside a function called "main". A function is simply a collection of commands that do "something". The main function is always called when the program first executes. From main, we can call other functions, whether they be written by us or by others or use built-in language features. To access the standard functions that comes with your compiler, you need to include a header with the #include directive. What this does is effectively take everything in the header and paste it into your program. Let's look at a working program:

```
#include <studio.h>
int main()
{
printf( "I am alive!  Beware.\n" );
getchar();
return 0;
}
```

Let's look at the elements of the program. The #include is a "preprocessor" directive that tells the compiler to put code from the header called studio.h into our program before actually creating the executable. By including header files, you can gain access to many different functions--both the printf and getchar functions are included in studio.h.

The next important line is intmain(). This line tells the compiler that there is a function named main, and that the function returns an integer, hence int. The "curly braces" ({ and }) signal the beginning and end of functions and other code blocks. If you have programmed in Pascal, you will know them as BEGIN and END. Even if you haven't programmed in Pascal, this is a good way to think about their meaning.

The printf function is the standard C way of displaying output on the screen. The quotes tell the compiler that you want to output the literal string as-is (almost). The '\n' sequence is actually treated as a single character that stands for a newline (we'll talk about this later in more detail); for the time being, just remember that there are a few sequences that, when they appear in a string literal, are actually not displayed literally by printf and that '\n' is one of them. The actual effect of '\n' is to move the cursor on your screen to the next line. Notice the semicolon: it tells the compiler that you're at the end of a command, such as a function call. You will see that the semicolon is used to end many lines in C.

The next command is getchar(). This is another function call: it reads in a single character and waits for the user to hit enter before reading the character. This line is included because many compiler environments will open a new console window, run the program, and then close the window before you can see the output. This command keeps that window from closing because the program is not done yet because it waits for you to hit enter. Including that line gives you time to see the program run.

Finally, at the end of the program, we return a value from main to the operating system by using the return statement. This return value is important as it can be used to tell the operating system whether our program succeeded or not. A return value of 0 means success. The final brace closes off the function. You should try compiling this program and running it. You can cut and paste the code into a file, save it as a .c file, and then compile it. If you are using a command-line compiler, such as Borland C++ 5.5, you should read the compiler instructions for information on how to compile. Otherwise compiling and running should be as simple as clicking a button with your mouse (perhaps the "build" or "run" button).

Features of C

C is a MIDDLE LEVEL LANGUAGE because it combines the elements of HIGH LEVEL LANGUAGE with the functionalism of ASSEMBLY LANGUAGE .C allows manipulation of bits, bytes and addresses, which are the basic elements with which the computer functions. Also, C code is very portable, that is software written on one type of computer can be adapted to work on another type. Although C has 5 basic built-in data types, it is not strongly typed language compared to high-level languages. C permits almost all data type conversions.

C is an ideal for system-level programming. The term block structure language does not apply strictly to C. Technically a block structure language permits procedure and functions to be declared inside other procedures of functions.

C does not allow the creation of functions within functions, and therefore cannot be formally called a block-structured language.

C allows compartmentalization of code and data. This is a distinguishing feature of any structured language. It refers to the ability of a language to section of and hides all information and instructions necessary to perform a specific task from the rest of the program. Code can be compartmentalized in C using functions or code blocks. FUNCTIONS are used to define and code separately, special tasks required in the program. This allows program to be modular. CODE BLOCK is a logically connected statement that is treated like a unit. Placing a sequence of between opening and closing flower braces creates a code block. Also a structured language supports several loop constructs, such as while, do-while, and for. It allows the user to indent statements and does not require a field concept. C was initially used for system programming. A system program forms a portion of the operating system of the computer of its support utilities. Operating system, Interpreters, Editors, Assembly programs are usually called system program.

The UNIX Operating System was developed using C. Today C programming used by many programmers to program all kind of task to its portability and efficiency. There is C compilers available for almost all computers. A code written in C on a particular machine can compiled and run on another machine by making a few or no changes. Also C compiler produces very fast object code. In addition to the above, C offers the speed of an assembly language, but few of the restrictions of a structured language. Programmers can create and maintain a unique library of functions, which can be used by many different programs. Thus large projects can be managed easily, with minimal duplication of effort.

Introduction of Graphics:

In today’s world of advance technology, interactive computer graphics has become a powerful tool for production of realistic features. Today, we find computer graphics used in various areas that include science, engineering, medicine, industry, art, entertainment etc. The main reason for the effectiveness of computer graphics is the speed with which the user can understand the displayed information. The graphics in Turbo C provides a wide variety of built in functions. Computer Graphics remains one of the most exciting and rapidly growing computer fields. It has become a common element in the user interface, data visualization, TV commercials, motion pictures and many other applications.

The current trend of computer graphics is to incorporate more physics principles into 3-D graphics algorithm to better simulate the complex interactions between objects and lighting environment.

Turbo C includes an extensive collection of graphics-oriented library functions. These functions fall into two categories:

1. Those that work in text mode.
2. Those that work in graphics mode

Text Mode Function:

- The text mode functions are concerned with placing text in certain areas of screen.
- They work in any graphics monitor and adapter.
- In this mode, the screen is divided into character positions, typically 80 columns and 25 rows.
- It also facilitates writing text to windows.

Graphics Mode Function:

- Graphics mode requires a graphics monitor and adapter card such as CGA, EGA and VGA.
- It allows one to draw dots, lines, and shapes, add colors to lines and areas and perform many other graphics related activities.
- In graphics mode, one can address individual pixels, which gives a much finer resolution.
- When we start drawing any graphics on the screen we need the header file called GRAPHICS.H and library file called GRAPHICS.LIB
- The header file contains definitions and explanations of all the functions and explanation of all the functions and constants needed in the program, and the graphic function are kept in the graphics library file. Both these functions are kept in the graphics library file. Both these files provide as a part of Turbo C++ package.
- To switch to over graphics mode that offers the best resolution, the initgraph() function is called. It figures out the best resolution and puts the corresponding number to that mode in the variable gm. The gm number tells the monitor the resolution we are using, the number of video pages it supports and the colors that are available.
- A variable gd can be assigned to DETECT, which asks the initgraph() to figure out the BGI file needed and load it into the memory.

- When switched to the graphics mode, firstly the cursor disappears since the graphics mode doesn’t support the conventional cursor.
- Secondly, the co-ordinate system is established whereby the top-left corner of the screen is treated as origin (0, 0). The X-axis goes horizontally across, and the Y-axis goes vertically downwards.
- In the graphics mode users can use functions like line(), circle(), rectangle() etc for drawing and functions like setfillstyle(), setcolor(), floodfill() can be used to fill the colors.
- Turbo C also provides generation of sounds using three functions namely sound(), delay() and nosound(). These files are defined in the DOS.H header file.
- When exiting from graphics mode the system should be restored to the previous video mode. To restore the previous mode abd release the memory used by the graphics mode, the closegraph() library function should be executed.

By 1960, a hoard of computer languages had come into existence, almost each for specific purposes. COBOL was being used for commercial applications. FORTRAN for Engineering and Scientific applications and so on. Instead of learning and using so many languages, each for a different purpose, an international committee was set up to develop a language which can program all possible applications. This committee came out with a language called ALGOL 60, which was too abstract and general. BCPL was developed by Martin Richards aimed to solve the problems of the previous languages. Around same time ‘B’ was developed by Ken Thomson at AT&T’s Bell Labs. Finally ‘C’ was originally developed in 1970’s by Dennis Ritchie at Bell Laboratories, Inc. It is an outgrowth of BCPL and B.

Graphics Primitives

A primitive is a graphics object that is essential for the creation or construction of complex images. Fortunately, graphics is constructed from three basic elements, as opposed to the great variety of graphics applications. The most basic of these elemental structures is the pixel, short for picture element.

Pixel:

A pixel is a point of light. It is just one tiny dot on the raster displays. Though it has no structure, it is definitely a building block and hence it can be considered as the graphics primitive.

The resolution of CRT is related to the dot size, the diameter of a single dot. A resolution of 100 dots lines/inch implies a dot size of 0.01 inch. However, in reality, pixels are more elliptic than circle. The shape of a pixel purely depends upon the characteristics of the visual display unit. The ratio of the distance between the centres's of two adjacent horizontal pixels to that of the vertical ones is called the pixel ratio.

For example, if 100-pixel lines in x and y measure 6cm and 8cm respectively, then

Pixel ratio should be considered in line-generating algorithms.

Line:

Line, especially straight lines, constitutes an important building block of computer images. For example, Line is the basic building block of Line graphs, bar and pie charts, two and three-dimensional graphs of mathematical functions, engineering drawings and architectural plans.

In computer graphics, straight line is so basic in creating images that we call it a graphics primitive. Straight lines can be developed in two different ways. A structural method determines which pixels should be set before drawing the line; a conditional method tests certain conditions to find which pixel should be set next.

Image Types:

Two-dimensional:

2D computer graphics are the computer-based generation of digital images—mostly from two-dimensional models, such as 2D geometric models, text, and digital images, and by techniques specific to them.

2D computer graphics are mainly used in applications that were originally developed upon traditional printing and drawing technologies, such as typography, cartography, technical drawing, advertising, etc.. In those applications, the two-dimensional image is not just a representation of a real-world object, but an independent artifact with added semantic value; two-dimensional models are therefore preferred, because they give more direct control of the image than 3D computer graphics, whose approach is more akin to photography than to typography.

Three-dimensional:

3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images. Such images may be for later display or for real-time viewing.

Despite these differences, 3D computer graphics rely on many of the same algorithms as 2D computer vector graphics in the wire frame model and 2D computer raster graphics in the final rendered display. In computer graphics software, the distinction between 2D and 3D is occasionally blurred; 2D applications may use 3D techniques to achieve effects such as lighting, and primarily 3D may use 2D rendering techniques.

3D computer graphics are often referred to as 3D models. Apart from the rendered graphic, the model is contained within the graphical data file. However, there are differences. A 3D model is the mathematical representation of any three dimensional object. A model is not technically a graphic until it is visually displayed. Due to 3D printing, 3D models are not confined to virtual space. A model can be displayed visually as a two-dimensional image through a process called 3D rendering, or used in non-graphical computer simulations and calculations. There is some 3D computer graphics software for users to create 3D images.

Graphics in C

Computer graphics is one of the most powerful and interesting facets of computers. All video games, animations, multimedia predominantly works in graphics. The available graphic modes in Turbo c are:

BGI graphics driver

Constant value

DETECT - 0(REQUESTS AUTO DETECTION)

CGA – 1

MCGA – 2

EGA – 3

EGA64 – 4

EGAMONO – 5

IBM8514 – 6

ERCMONO – 7

ATT400 – 8

VGA – 9

PC3270 – 10

In c, graphics can be implemented easily. The header fileGraphics' and library file Graphics.lib contains definitions and explanations of all the constants and functions we need.

Four VGA modes are available:

640*200(16 color)

640 * 350(16 color)

640 * 480(16 color)

800*600 with 256 color