Overview:

C programming language is well developed since 1972. C Compiler appears in all operating systems. Usually they are with free public license, i.e. for use For example GNU C project or gcc. Traditionally C code has to be written in text editor and then compiled, where the compiler is in fact optimized for the particular hardware platform and OS environment. For example the outcome of UNIX based compiling “a.out” cannot run on Windows OS. Hence the compiler is specific with regard the computer and operating system. In present days the complexity of the most advanced C projects makes this traditional procedure of writing code and compiling somehow troublesome in case when we need to deal with hundreds of subroutines or/and debugging thousands of lines of code. Therefore in order to speedup the programming process, integrated development environment (IDE) were developed. Essentially IDE is a software application, which provides comprehensive facilities to computer programmers for software development. There are many IDEs for almost all high level programming languages and most of them are commercial. For C (C++) development in Windows environment Microsoft distributes Visual Studio which is the topic of this supplement material

Programming

Suppose you are about to develop algorithm in your favorite language which for now we assume is C. However you are interested to develop it as fast as possible, from the writing, testing and debugging. Most likely you will run Microsoft Windows operating system which makes quite possible for you to have at hand freely distributed Visual Studio Express (VSE) edition package. This is Microsoft IDE product for fast development on several languages including C/C++. For the purpose of this handout we are going to use the same program that you have successfully debugged in Lab 2 assignment. We will show how to start the VSE how to compile and run your project. Finally we will show how to debug your code.

1. Start Visual C++ Studio Express Edition

2. Create a new project by clicking File->New->Project. You will see the screen on figure 1.

3. Chose Project Win32 on the left hand side. Then pick Win32 console application, give a name ‘test” and click OK

4. On the next screen specify that the project will be empty and click Finish (figure 2).

5. Now we have created a project called Test. Keep in your mind that this is not yet a code just we started a project potentially with many codes inside. The next step is to add code to our project. Click on Source file with right mouse button : Add->New Item. The screen should look as it is shown on figure 3.

6. As soon as you add new item you have to specify what we would like to add. The available options are shown on the figure 4.
7. Now chose that we add code on the left screen side and pick C++ code. Give a name in the Name field on the bottom: “code1”. Now you will see empty code space as it is shown on figure 5.

8. If you have at hand the code1.c from the Lab2 assignment, please open, copy and paste the contents in the newly created code1 in VSE (figure 6).

9. From the Debug menu choose Start Without Debugging. You will be asked whether you want to build the project. Choose “yes”. Any errors in your code will be displayed in the bottom window. By click on these messages you will navigate through the errors.

**Debugging**

10. As we know the code has 3 bugs. In order to determine the bugs we place a breakpoint by clicking with the mouse on the left side the code. A red ball will appear.

11. The debugging can be started by pressing F5 and the execution will halt on the breakpoint. The stepping mode through the code is possible by pressing F11 instead of F5. All different modes are accessible in the menu of Debug on the top menu bar.

12. We would like to watch some variables. For example the local variable numin in GetGloat() procedure is taking wrong values. We can add this variable in the watch-window on the bottom left as going Debug-Quick Watch, then type the name of the variable and press Add Watch button. On the figure 7 we show the execution of the code just after assigning the value 13 from the keyboard. We can see that we assigned 13 from the keyboard, but numin hold value 1.82e-44, which denotes a bug in Getfloat() procedure.

13. After successful debugging and correcting all errors you may want to keep your executable code. Since we started a project, where we added a single file in order to debug, the final version is located in “…My Documents\Visual Studio 2005\Projects\test\debug’” folder. Screen caption on already debugged application is shown on the figure 8:

**8 Queen Problem**

In the following section we will take a close look at the 8 queen problem. The goal of the assignment is to write a code for finding all solution of the 8 queen problem from specified initial board. The solution of this problem must implement back track algorithm. Let’s take a look at it.

Consider the naïve approach of placing a queen on the board. Initially the board is empty and we will assume the convention of placing queen at the columns from left to right and on the rows from bottom up. Hence since A1 is not occupied we can place a queen there. Next we no longer will consider the first column since it is already occupied. Now we start with the second column again from bottom up manner. B1 cannot be taken, neither B2. Why? Simply because B1 is clash on horizontal and B2 is clash on diagonal. The first available position is B3. In similar manner for the third column we find the first available position to be C5. Now let’s analyze the moves. For each column where we are about to place a queen we start from the first row and check for:

1. Horizontal clash: we only have to check if there are queens on the left of the column on the same row.

2. Diagonal clash: Since there are two diagonals for a row i at the column j we need to check for queen at position (i-1) and (i+1) for column (j-1). Next we do
for (i-2) and (i+2) for column (j-2) and so on until we reach the end of the board in both directions.

There is no need to check the diagonals on the right of the position neither the vertical position for clashes since we place one queen at a column.

Now suppose we have the situation depicted on the following figure:

![Diagram of the eight queen problem](image)

Simply there is no place for the eight queen. Than we need to backtrack. To understand the backtrack algorithm we should keep in mind that the goal is to place all queens on the board. If for some particular board setup there is no way for queen placing, than we must go back and change the board in some way. Which way is the best? Obviously if we change the queen at the first column we need to rearrange the entire board. Short observation will suggest that we only can change the last placed queen and then review the problematic 8th column. But we placed the 7th queen by starting from row 1 until we reached row 6. Hence we if have do relocate it we shall consider only the rows above the 6th.

Now we shall see that there is no place for the seventh queen on any other rows except the 6th which we decided to backtrack. Therefore we conclude that there is no solution for the 6th queen and we have to do backtrack again. Does it look familiar? Yes this can be encoded with recursive function.

Finally, the backtrack algorithm can be described as following pseudo code:

```c
int put_queen(int column) /*Put queen on column y*/
{
    if (column>8)
        return done
    for (row = 1; row <= 8;row++) /*Start from the first row*/
    {
        if (isSafe (column,row)) /*Check if this position is safe*/
            {  
                keep_queen(column,row);
                if (put_queen(column+1))
                    return done
            }
        else
            remove_queen(column,row)
    }
    if (y<8) /*Done?*/
        put_queen(y+1); /*Put queen on the next column*/
    else
        print_board() /*Print board*/
}
```
Lab Assignment

Implement algorithm for queen placement and write a function GetQueenPosition. The following code can be used for this assignment.

```c
#include <stdio.h> /* for scanf, printf, etc ... */
int GetQueenPosition(int board[8], int row)
{
    /*
    Your code comes here
    */
}
int main(void)
{
    /*initialize empty board*/
    int board[8] = {0,0,0,0,0,0,0,0};
    int k;

    /*Find queen position for column 1*/
    k = GetQueenPosition(board, 0);
    printf("A queen can be placed at row \%d of column 1\n", k);

    /*Find queen position for column 2*/
    board[0] = 1;
    k = GetQueenPosition(board, 0);
    printf("A queen can be placed at row \%d of column 2\n", k);

    /*Find queen position for column 5*/
    board[1] = 7;
    board[2] = 4;
    board[3] = 6;
    k = GetQueenPosition(board, 0);
    printf("A queen can be placed at row \%d of column 5\n", k);

    /*Find queen position for column 5, above 3th row*/
    k = GetQueenPosition(board, 3);
    printf("A queen can be placed at row \%d of column 5\n", k);
}
```
Figure 1

Figure 2
Figure 7

```cpp
float GetFloat(float num)
{
    float numb;
    /* read in the float */
    printf("Enter number: ", num);
    scanf("%f", &numb);
    /* return its value */
    return numb;
}
```

Figure 8

```
C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\Projects\test\debug\test
Enter number:1
Enter number:2
Enter number:3
Enter number:4
Enter number:5
Enter number:6
Enter number:7
Enter number:8
Enter number:9
Enter number:10
Enter number:0
The product is 3628800.000000
```

C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\Projects\test\debug