Convert ASCII (TXT) File to Binary File

```
#include <cstdio>

#define DataNum 102

struct SudokuDataHeader {
    int numbers;
    int datasize;
};

struct SudokuProblem {
    int id;
    int data[9][9];
};

int main()
{
    FILE *fin, *fout;
    SudokuDataHeader sh;
    SudokuProblem sp;

    fin = fopen("all.txt", "r");
    fout = fopen("sudoku.dat", "wb");
    sh.numbers = DataNum;
    sh.datasize = sizeof(SudokuProblem);
    fwrite((void*)&sh, sizeof(SudokuDataHeader), 1, fout);

    char c;
    int  row, col;
    sp.id = 1;
    row=col=0;
    while (fscanf(fin, "%c", &c)!=EOF) {
        if (c=='.') sp.data[row][col++] = 0;
        else if (c>='0'&&c<='9') sp.data[row][col++] = c-'0';
        if (col==9) { row++; col=0; }
        if (row==9) {
            printf("Finish problem %d\n", sp.id);
            row=col=0;
            fwrite((void*)&sp, sizeof(sp), 1, fout);
            sp.id++;
        }
    }
    fclose(fin);
    fclose(fout);
    return 0;
}
```

Random Choose a Problem and Solve it

```
#include <cstdio>
#include <cstdlib>
#include <ctime>

void print_board(int puzzle[][9]);
int solve(int puzzle[][9], int pos);
int isValid(int number, int puzzle[][9], int row, int col);

struct SudokuDataHeader {
    int numbers;
    int datasize;
};

struct SudokuProblem {
    int id;
    int data[9][9];
};

int main()
{
    FILE *fp;
    SudokuDataHeader sdh;
    SudokuProblem sp;

    fp = fopen("sudoku.dat", "rb");
    fread((void*)&sdh, sizeof(sdh),  1, fp); // Read header
    printf("\nTotal Problems = %d\n", sdh.numbers);
    printf("Each occupys %d bytes.\n\n", sdh.datasize);

    srand(time(NULL)); // Randomize the seed
    int k = rand()%sdh.numbers; // Select int from 0~numbers-1
    fseek(fp, k*sdh.datasize, SEEK_CUR); // Jump k records
    fread((void*)&sp, sizeof(sp), 1, fp);

    int count = 0;
    for (int i=0; i<9; i++) for (int j=0; j<9; j++) {
        if (sp.data[i][j]) count++;
    }
    printf("Problem ID: %d (Count=%d)\n", sp.id, count);
    print_board(sp.data);
    printf("\nSolution:\n");
    if (solve(sp.data, 0)) print_board(sp.data);
    else printf("No solution!");

    return 0;
}

int solve(int puzzle[][9], int pos)
{
    if (pos>80) return 1;

    int row=pos/9, col=pos%9;

    if (puzzle[row][col]) return (solve(puzzle, pos+1));

    for (int nextNum=1; nextNum<10; nextNum++) {
        if(isValid(nextNum, puzzle, row, col)) {
            puzzle[row][col] = nextNum;
            if (solve(puzzle, pos+1)) return 1;
        }
```

```
    }
    // Failed to find a valid value, go back to previous cell for another try
    puzzle[row][col] = 0; // reset before goback
    return 0;
}

int isValid(int number, int puzzle[][9], int row, int col)
{
    int rowStart = (row/3) * 3;
    int colStart = (col/3) * 3;

    for(int i=0; i<9; ++i)
    {
        if (puzzle[row][i] == number) return 0;
        if (puzzle[i][col] == number) return 0;
        if (puzzle[rowStart + (i%3)][colStart + (i/3)] == number) return 0;
    }
    return 1;
}

void print_board(int puzzle[][9])
{
    printf("\n +-------+-------+-------+\n");
    for(int i=1; i<10; ++i) {
        for(int j=1; j<10; ++j) {
            if (j%3==1) printf(" | ");
            else printf(" ");
            printf("%d", puzzle[i-1][j-1]);
        }
        printf(" |\n");
        if (i%3 == 0) printf(" +-------+-------+-------+\n");
    }
}
```