Names rules

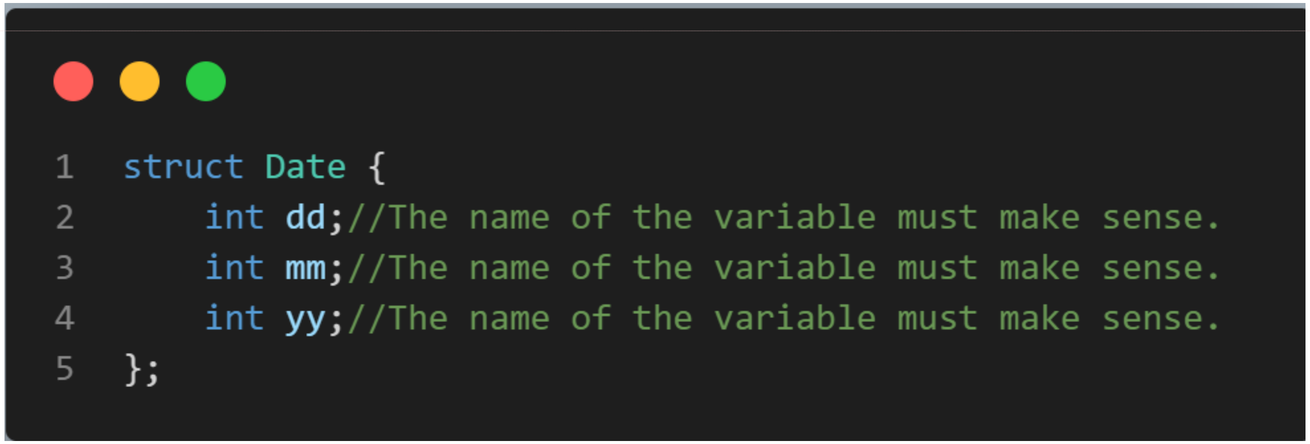
- Choose descriptive and unambiguous names.
- Make meaningful distinction.
- Use pronounceable names.
- Use searchable names.
- Replace magic numbers with named constants.
- Avoid encodings. Don't append prefixes or type information

Functions rules

1. Small.
2. Do one thing.
3. Use descriptive names.
4. Prefer fewer arguments.
5. Have no side effects.
6. Don't use flag arguments. Split method into several independent methods that can be called from the client without the flag.

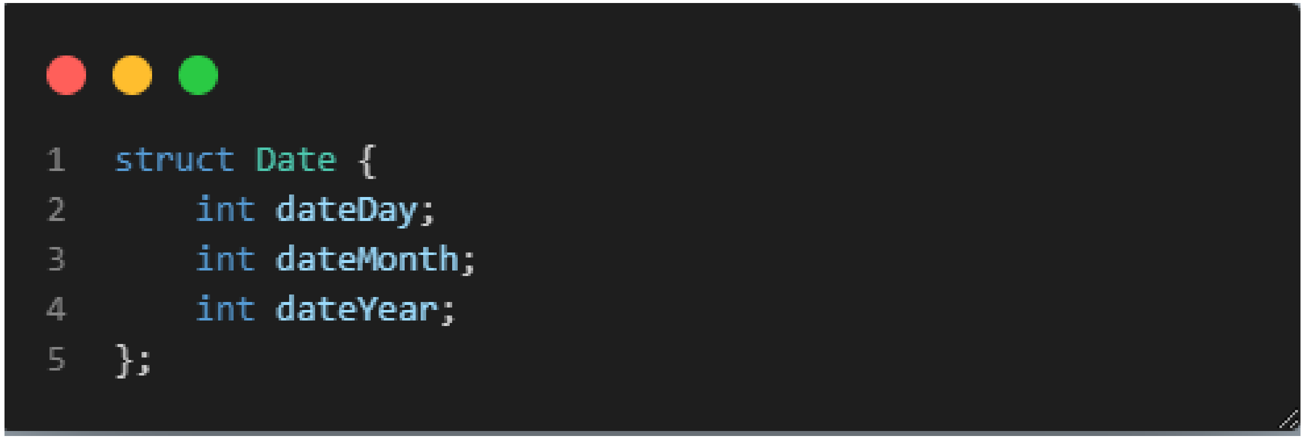
Comments rules

1. Always try to explain yourself in code.
2. Don't be redundant.
3. Don't add obvious noise.
4. Don't use closing brace comments.
5. Don't comment out code. Just remove.
6. Use as explanation of intent.
7. Use as clarification of code.
8. Use as warning of consequences

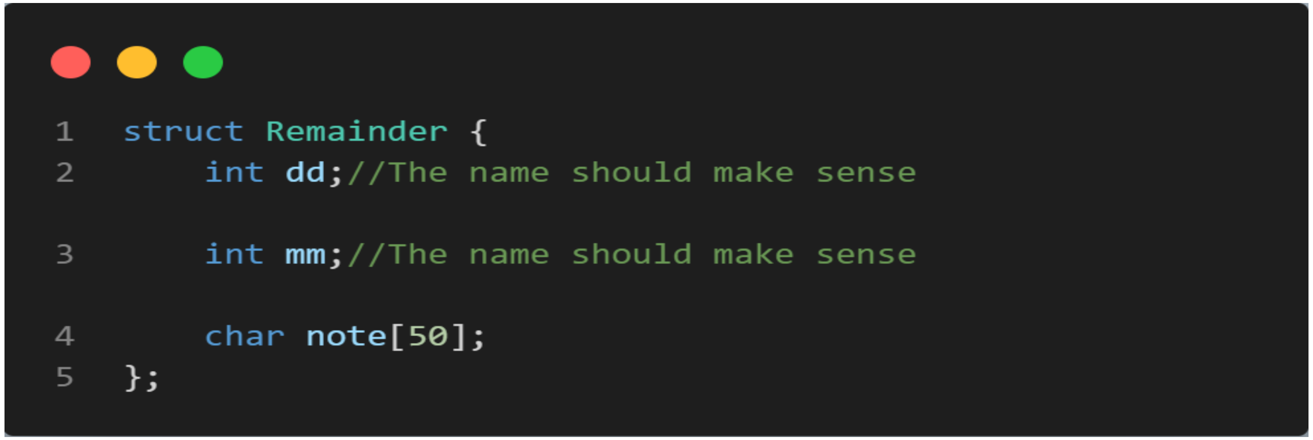


```
struct Date {
    int dd;//The name of the variable must make sense.
    int mm;//The name of the variable must make sense.
    int yy;//The name of the variable must make sense.
};
```

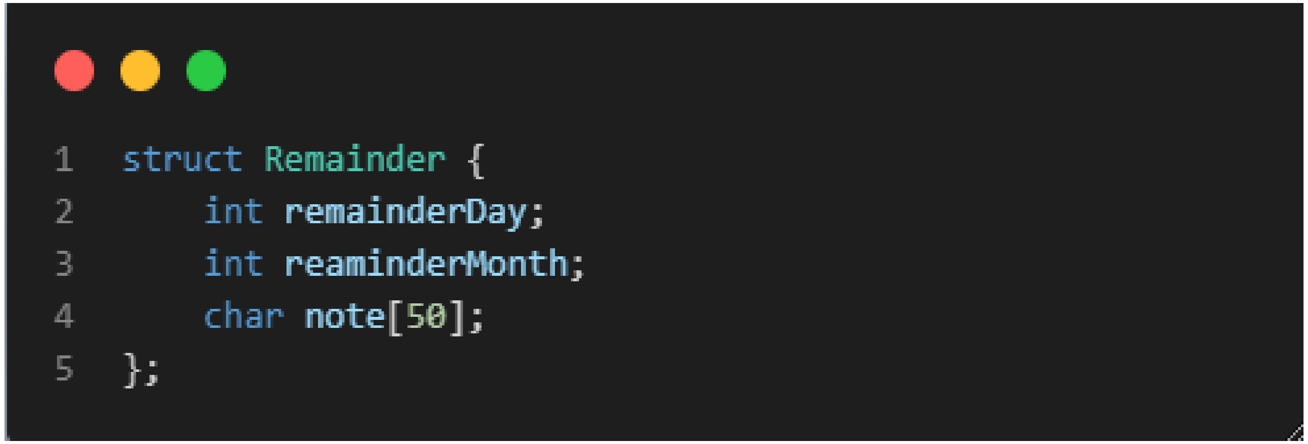
:Commented [y1]



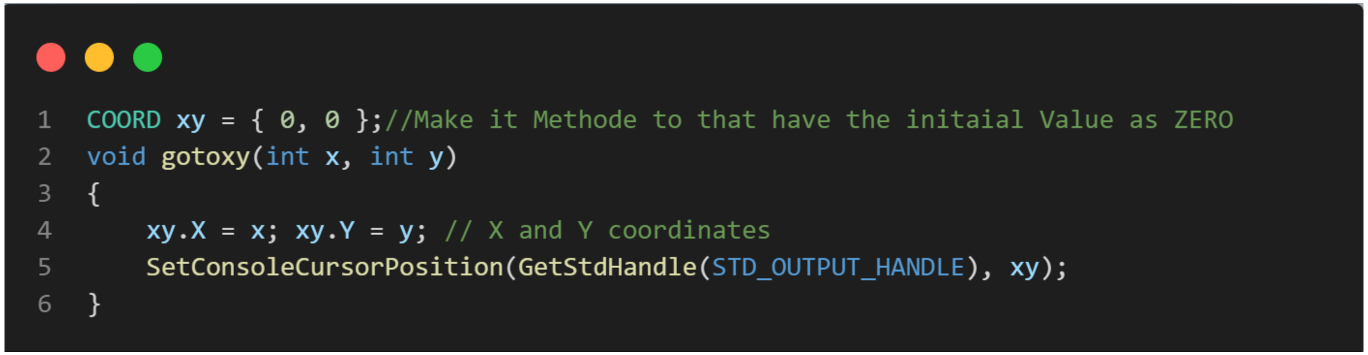
```
struct Date {
    int dateDay;
    int dateMonth;
    int dateYear;
};
```



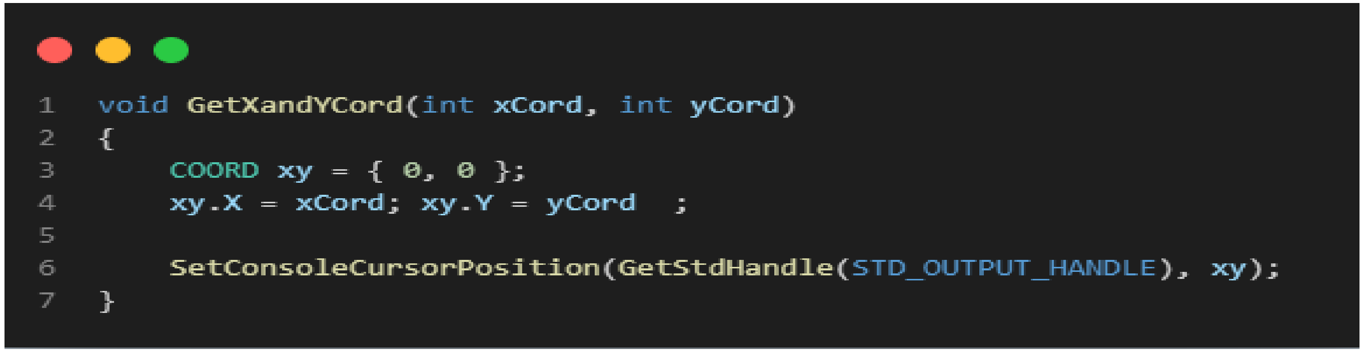
```
struct Remainder {
    int dd;//The name should make sense
    int mm;//The name should make sense
    char note[50];
};
```



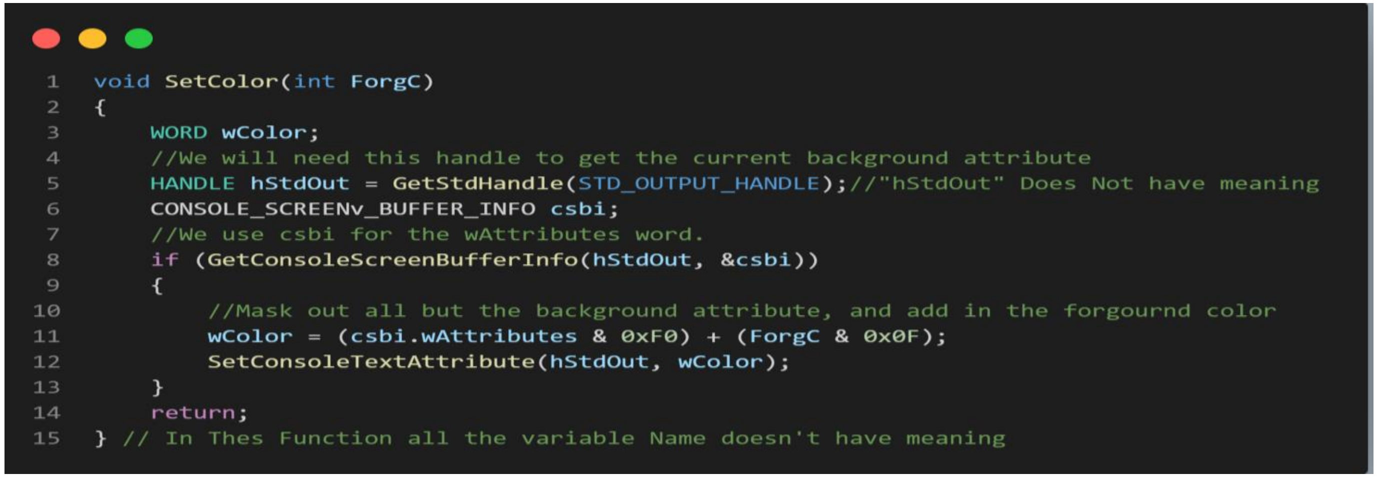
```
struct Remainder {
    int remainderDay;
    int reaminderMonth;
    char note[50];
};
```



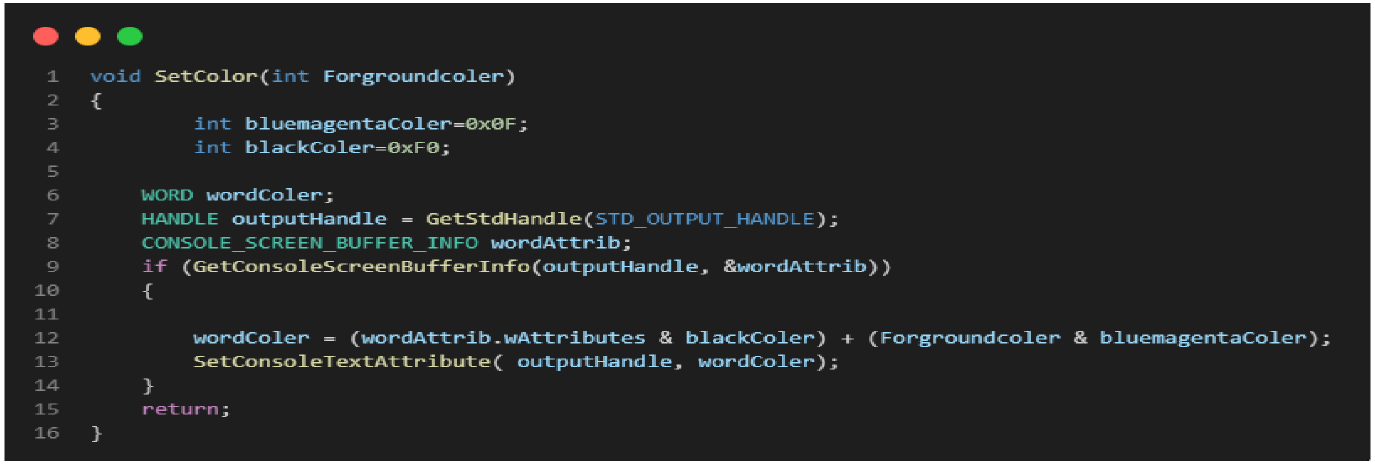
```
COORD xy = { 0, 0 };//Make it Methode to that have the initaial Value as ZERO
void gotoxy(int x, int y)
{
    xy.X = x; xy.Y = y; // X and Y coordinates
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), xy);
}
```



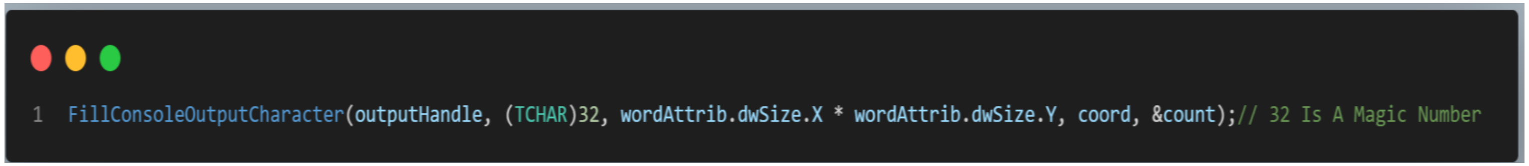
```
void GetXandYCord(int xCord, int yCord)
{
     COORD xy = { 0, 0 };
     xy.X = xCord; xy.Y = yCord   ;

     SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), xy);
}
```



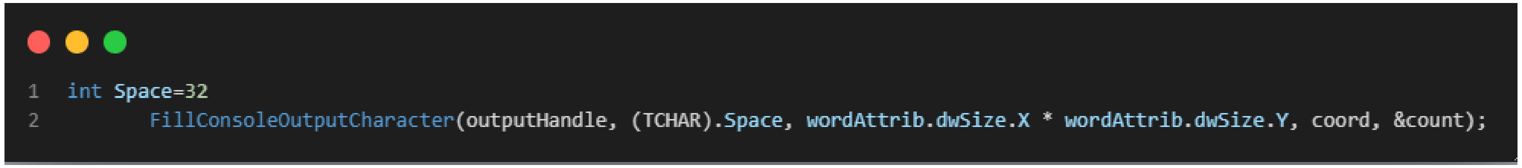
```
void SetColor(int ForgC)
{
    WORD wColor;
    //We will need this handle to get the current background attribute
    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);//"hStdOut" Does Not have meaning
    CONSOLE_SCREENv_BUFFER_INFO csbi;
    //We use csbi for the wAttributes word.
    if (GetConsoleScreenBufferInfo(hStdOut, &csbi))
    {
        //Mask out all but the background attribute, and add in the forgournd color
        wColor = (csbi.wAttributes & 0xF0) + (ForgC & 0x0F);
        SetConsoleTextAttribute(hStdOut, wColor);
    }
    return;
} // In Thes Function all the variable Name doesn't have meaning
```



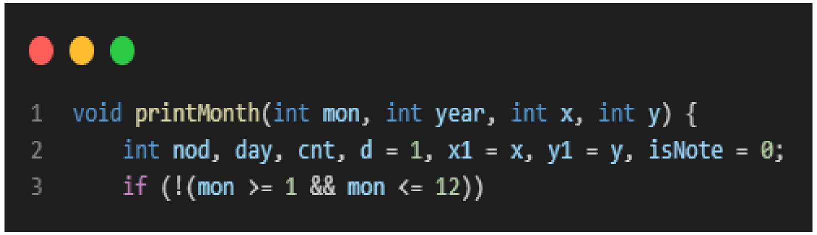
```
void SetColor(int Forgroundcoler)
{
        int bluemagentaColer=0x0F;
        int blackColer=0xF0;

    WORD wordColer;
    HANDLE outputHandle = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO wordAttrib;
    if (GetConsoleScreenBufferInfo(outputHandle, &wordAttrib))
    {

        wordColer = (wordAttrib.wAttributes & blackColer) + (Forgroundcoler & bluemagentaColer);
        SetConsoleTextAttribute( outputHandle, wordColer);
    }
    return;
}
```



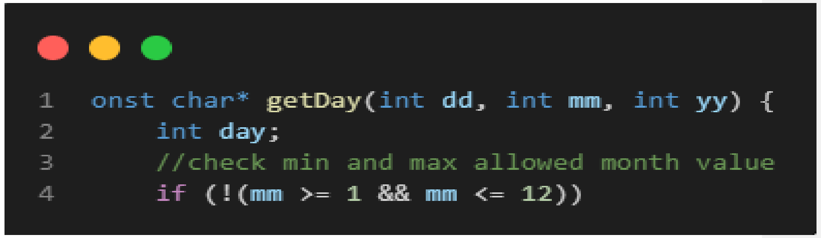
```
FillConsoleOutputCharacter(outputHandle, (TCHAR)32, wordAttrib.dwSize.X * wordAttrib.dwSize.Y, coord, &count);// 32 Is A Magic Number
```



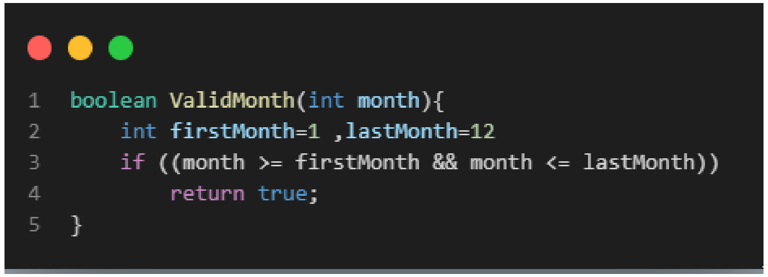
```
int Space=32
        FillConsoleOutputCharacter(outputHandle, (TCHAR).Space, wordAttrib.dwSize.X * wordAttrib.dwSize.Y, coord, &count);
```



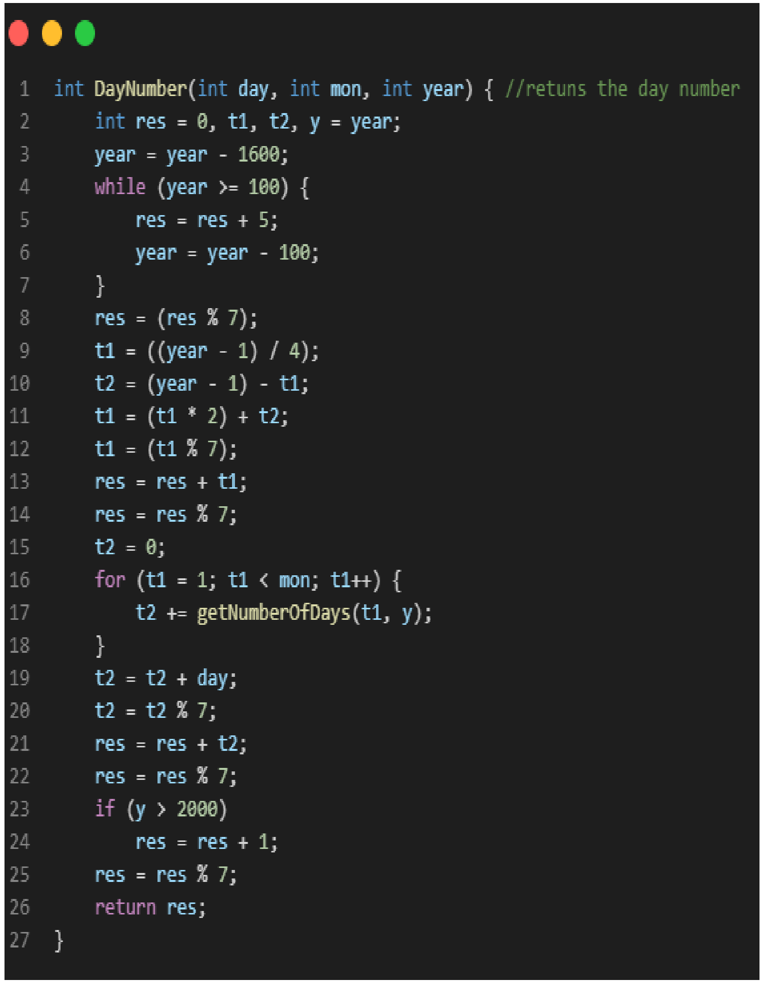
```
void printMonth(int mon, int year, int x, int y) {
    int nod, day, cnt, d = 1, x1 = x, y1 = y, isNote = 0;
    if (!(mon >= 1 && mon <= 12))
```



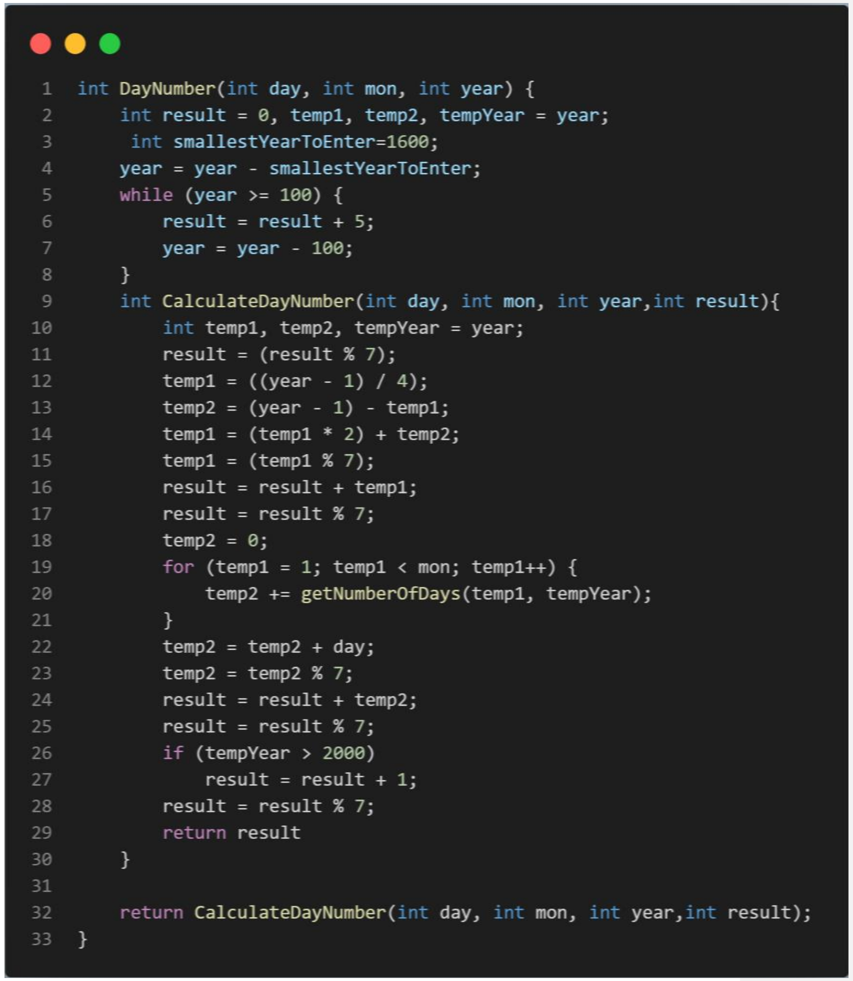
```
onst char* getDay(int dd, int mm, int yy) {
    int day;
    //check min and max allowed month value
    if (!(mm >= 1 && mm <= 12))
```



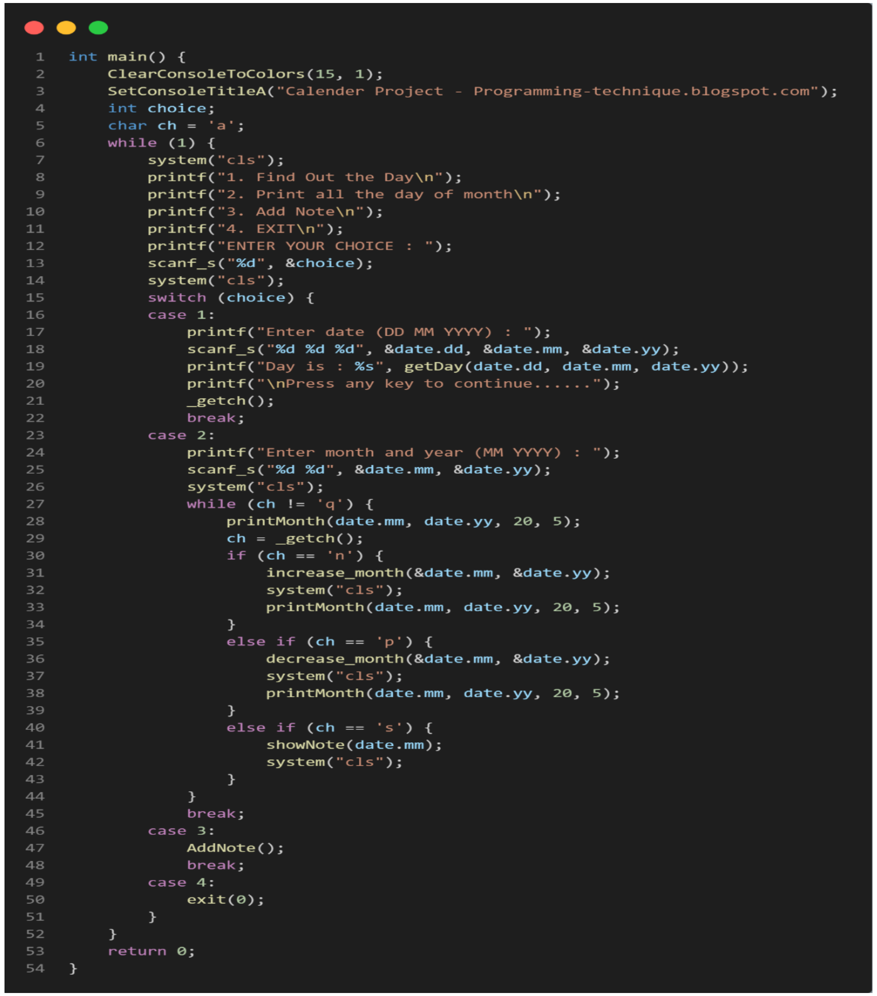
```
boolean ValidMonth(int month){
    int firstMonth=1 ,lastMonth=12
    if ((month >= firstMonth && month <= lastMonth))
        return true;
}
```



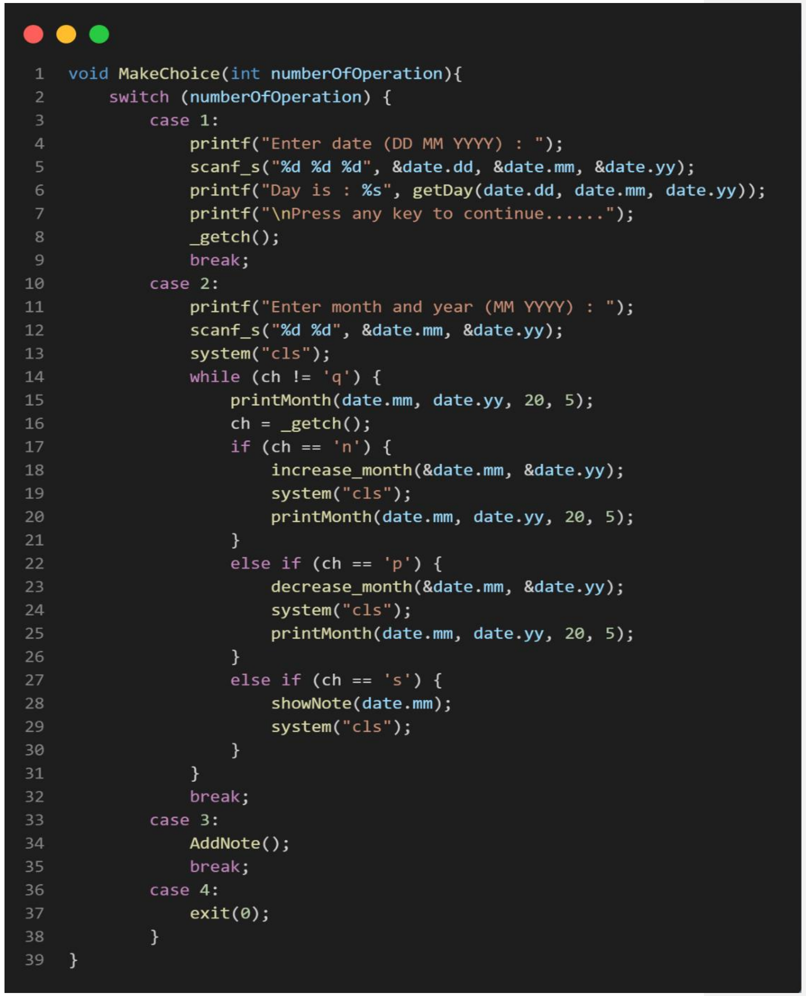
```
int DayNumber(int day, int mon, int year) { //retuns the day number
    int res = 0, t1, t2, y = year;
    year = year - 1600;
    while (year >= 100) {
        res = res + 5;
        year = year - 100;
    }
    res = (res % 7);
    t1 = ((year - 1) / 4);
    t2 = (year - 1) - t1;
    t1 = (t1 * 2) + t2;
    t1 = (t1 % 7);
    res = res + t1;
    res = res % 7;
    t2 = 0;
    for (t1 = 1; t1 < mon; t1++) {
        t2 += getNumberOfDays(t1, y);
    }
    t2 = t2 + day;
    t2 = t2 % 7;
    res = res + t2;
    res = res % 7;
    if (y > 2000)
        res = res + 1;
    res = res % 7;
    return res;
}
```



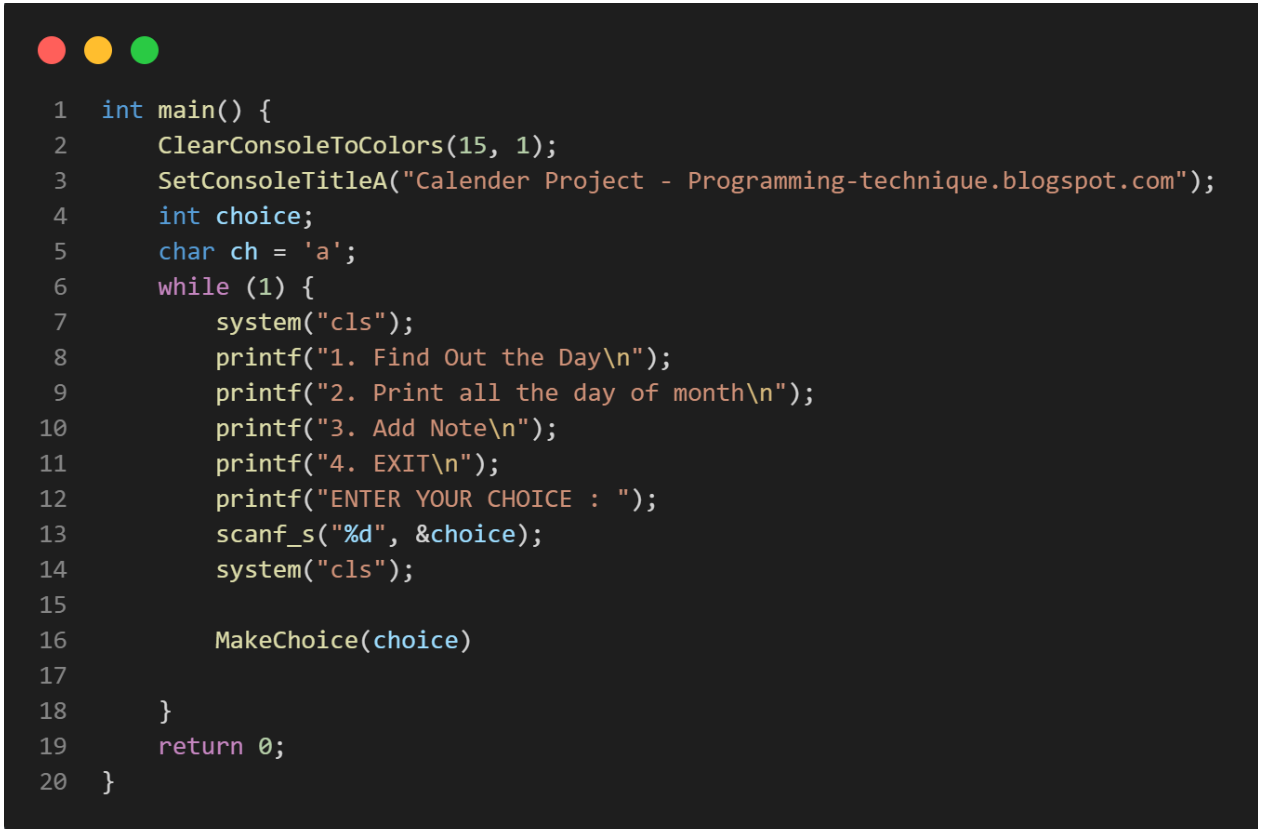
```
int DayNumber(int day, int mon, int year) {
    int result = 0, temp1, temp2, tempYear = year;
     int smallestYearToEnter=1600;
    year = year - smallestYearToEnter;
    while (year >= 100) {
        result = result + 5;
        year = year - 100;
    }
    int CalculateDayNumber(int day, int mon, int year,int result){
        int temp1, temp2, tempYear = year;
        result = (result % 7);
        temp1 = ((year - 1) / 4);
        temp2 = (year - 1) - temp1;
        temp1 = (temp1 * 2) + temp2;
        temp1 = (temp1 % 7);
        result = result + temp1;
        result = result % 7;
        temp2 = 0;
        for (temp1 = 1; temp1 < mon; temp1++) {
            temp2 += getNumberOfDays(temp1, tempYear);
        }
        temp2 = temp2 + day;
        temp2 = temp2 % 7;
        result = result + temp2;
        result = result % 7;
        if (tempYear > 2000)
            result = result + 1;
        result = result % 7;
        return result
    }

    return CalculateDayNumber(int day, int mon, int year,int result);
}
```



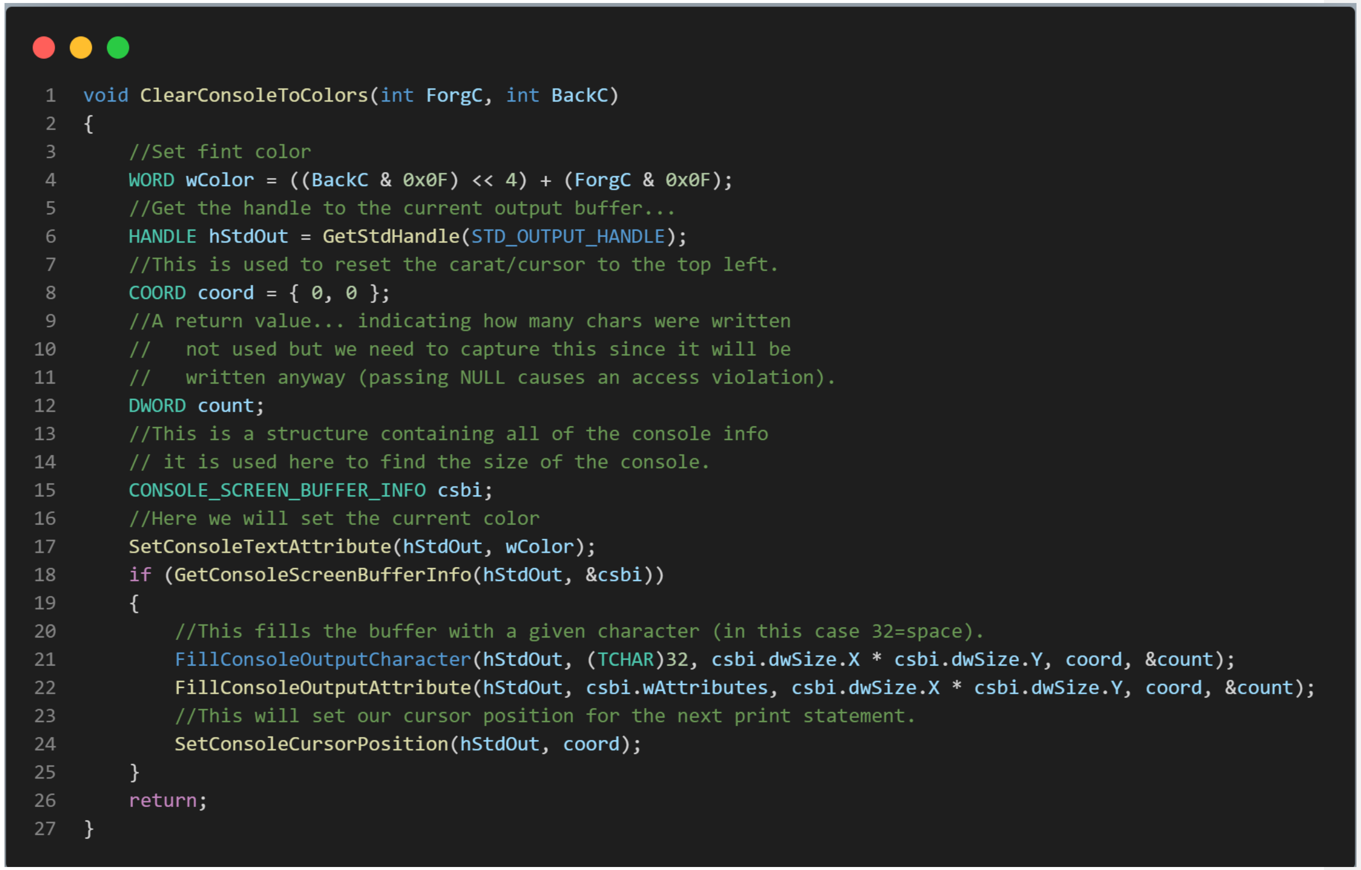
```c
int main() {
    ClearConsoleToColors(15, 1);
    SetConsoleTitleA("Calender Project - Programming-technique.blogspot.com");
    int choice;
    char ch = 'a';
    while (1) {
        system("cls");
        printf("1. Find Out the Day\n");
        printf("2. Print all the day of month\n");
        printf("3. Add Note\n");
        printf("4. EXIT\n");
        printf("ENTER YOUR CHOICE : ");
        scanf_s("%d", &choice);
        system("cls");
        switch (choice) {
        case 1:
            printf("Enter date (DD MM YYYY) : ");
            scanf_s("%d %d %d", &date.dd, &date.mm, &date.yy);
            printf("Day is : %s", getDay(date.dd, date.mm, date.yy));
            printf("\nPress any key to continue......");
            _getch();
            break;
        case 2:
            printf("Enter month and year (MM YYYY) : ");
            scanf_s("%d %d", &date.mm, &date.yy);
            system("cls");
            while (ch != 'q') {
                printMonth(date.mm, date.yy, 20, 5);
                ch = _getch();
                if (ch == 'n') {
                    increase_month(&date.mm, &date.yy);
                    system("cls");
                    printMonth(date.mm, date.yy, 20, 5);
                }
                else if (ch == 'p') {
                    decrease_month(&date.mm, &date.yy);
                    system("cls");
                    printMonth(date.mm, date.yy, 20, 5);
                }
                else if (ch == 's') {
                    showNote(date.mm);
                    system("cls");
                }
            }
            break;
        case 3:
            AddNote();
            break;
        case 4:
            exit(0);
        }
    }
    return 0;
}
```



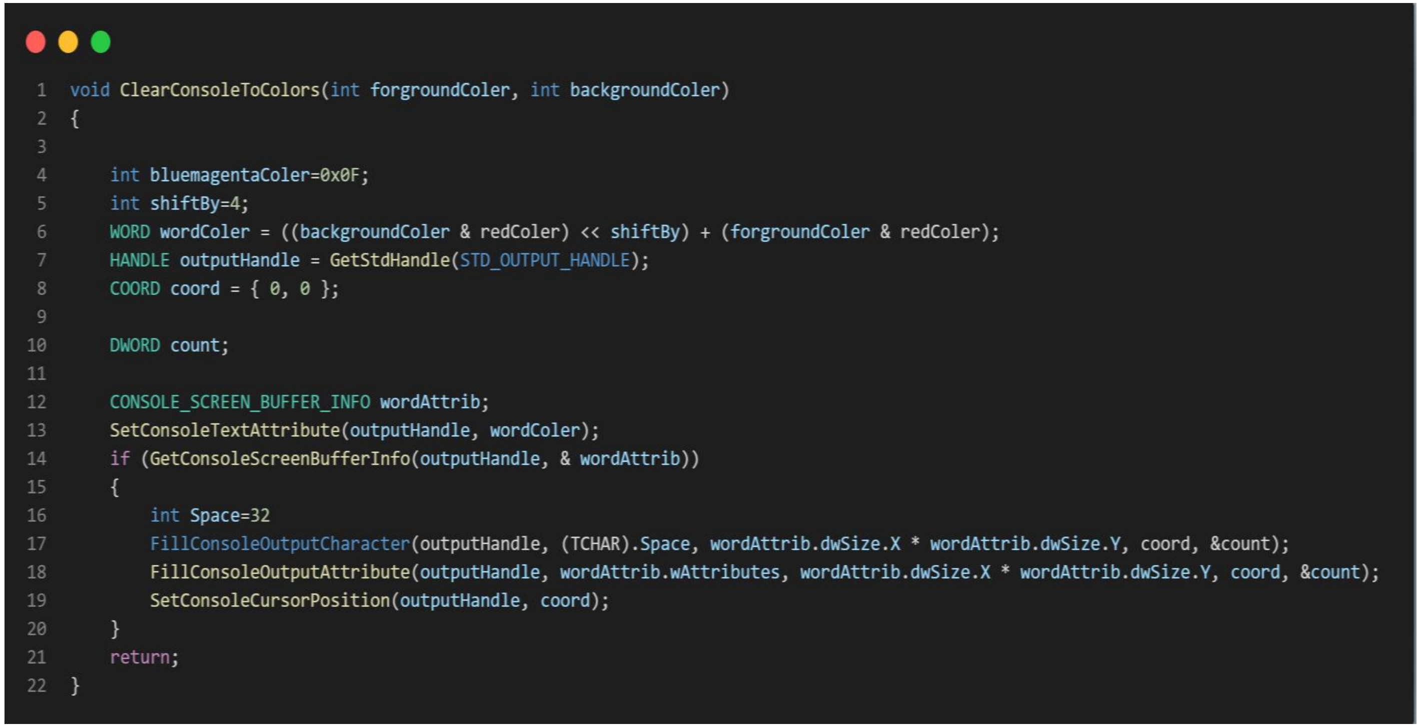
```c
void MakeChoice(int numberOfOperation){
    switch (numberOfOperation) {
        case 1:
            printf("Enter date (DD MM YYYY) : ");
            scanf_s("%d %d %d", &date.dd, &date.mm, &date.yy);
            printf("Day is : %s", getDay(date.dd, date.mm, date.yy));
            printf("\nPress any key to continue......");
            _getch();
            break;
        case 2:
            printf("Enter month and year (MM YYYY) : ");
            scanf_s("%d %d", &date.mm, &date.yy);
            system("cls");
            while (ch != 'q') {
                printMonth(date.mm, date.yy, 20, 5);
                ch = _getch();
                if (ch == 'n') {
                    increase_month(&date.mm, &date.yy);
                    system("cls");
                    printMonth(date.mm, date.yy, 20, 5);
                }
                else if (ch == 'p') {
                    decrease_month(&date.mm, &date.yy);
                    system("cls");
                    printMonth(date.mm, date.yy, 20, 5);
                }
                else if (ch == 's') {
                    showNote(date.mm);
                    system("cls");
                }
            }
            break;
        case 3:
            AddNote();
            break;
        case 4:
            exit(0);
        }
}
```



```c
int main() {
    ClearConsoleToColors(15, 1);
    SetConsoleTitleA("Calender Project - Programming-technique.blogspot.com");
    int choice;
    char ch = 'a';
    while (1) {
        system("cls");
        printf("1. Find Out the Day\n");
        printf("2. Print all the day of month\n");
        printf("3. Add Note\n");
        printf("4. EXIT\n");
        printf("ENTER YOUR CHOICE : ");
        scanf_s("%d", &choice);
        system("cls");

        MakeChoice(choice)

    }
    return 0;
}
```



```
void ClearConsoleToColors(int ForgC, int BackC)
{
    //Set fint color
    WORD wColor = ((BackC & 0x0F) << 4) + (ForgC & 0x0F);
    //Get the handle to the current output buffer...
    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    //This is used to reset the carat/cursor to the top left.
    COORD coord = { 0, 0 };
    //A return value... indicating how many chars were written
    //   not used but we need to capture this since it will be
    //   written anyway (passing NULL causes an access violation).
    DWORD count;
    //This is a structure containing all of the console info
    // it is used here to find the size of the console.
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    //Here we will set the current color
    SetConsoleTextAttribute(hStdOut, wColor);
    if (GetConsoleScreenBufferInfo(hStdOut, &csbi))
    {
        //This fills the buffer with a given character (in this case 32=space).
        FillConsoleOutputCharacter(hStdOut, (TCHAR)32, csbi.dwSize.X * csbi.dwSize.Y, coord, &count);
        FillConsoleOutputAttribute(hStdOut, csbi.wAttributes, csbi.dwSize.X * csbi.dwSize.Y, coord, &count);
        //This will set our cursor position for the next print statement.
        SetConsoleCursorPosition(hStdOut, coord);
    }
    return;
}
```



```
void ClearConsoleToColors(int forgroundColer, int backgroundColer)
{

    int bluemagentaColer=0x0F;
    int shiftBy=4;
    WORD wordColer = ((backgroundColer & redColer) << shiftBy) + (forgroundColer & redColer);
    HANDLE outputHandle = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD coord = { 0, 0 };

    DWORD count;

    CONSOLE_SCREEN_BUFFER_INFO wordAttrib;
    SetConsoleTextAttribute(outputHandle, wordColer);
    if (GetConsoleScreenBufferInfo(outputHandle, & wordAttrib))
    {
        int Space=32
        FillConsoleOutputCharacter(outputHandle, (TCHAR).Space, wordAttrib.dwSize.X * wordAttrib.dwSize.Y, coord, &count);
        FillConsoleOutputAttribute(outputHandle, wordAttrib.wAttributes, wordAttrib.dwSize.X * wordAttrib.dwSize.Y, coord, &count);
        SetConsoleCursorPosition(outputHandle, coord);
    }
    return;
}
```