INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Redes de Computadoras

“IP Calculator”

Abstact

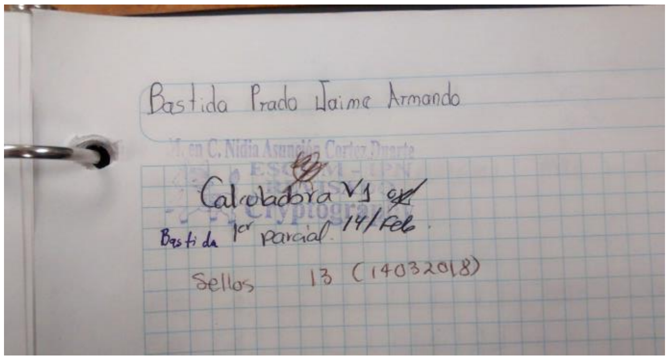
This practice consists of a IP calculator that determines the following characteristics of an IP: Class, Type, Network IP, Broadcast IP, and the Range of the host for that IP in specific.

By:

Bastida Prado Jaime Armando

Professor:
MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

March 2018


Bastida Prado Jaime Armando

M. en C. Nidia Asunción Cortez Duarte
ESCOM - IPN
REVISADO
Cryptography

Calculadora V1 ok
Bastida 1er parcial 14/Feb

Sellos 13 (14032018)

Index

Contenido

Introduction:

This program determines the most important properties of an IP Address given by the user, the results are calculated by operations over the bits of the IP using binary operators such as: “&” (and), “|” (or inclusive), “^” (or exclusive) and “~” (complement). The results are shown in console.

Literature review:

An IP Address consists of four bytes (each byte consisting of 8 bits), giving a maximum decimal value of 255 for each byte. Each byte is separated from the next by a dot, for example: “192.34.54.1”. The previous notation helps the reader identify different characteristics of an IP, this includes the Class, Type, Network Address, Broadcast Address and Range of host.

The Class is identified by the decimal value of the first byte in an IP, the existing classes are:

-A from 0 to 127

-B from 128 to 191

-C from 192 to 223

-D from 224 to 239

-E from 240 to 255

The Class A uses its first byte to denote the Network IP identifier and the rest three bytes are used by the Host. Class B uses its first two bytes for the Network IP and Class C uses its three first bytes.

A Network Mask is an Address used to identify the Network and Broadcast IP of a given IP Address, for now lets say that the Mask for Class A is “255.0.0.0”, for Class B “255.255.0.0” and C “255.255.255.0”.

A Network IP has all its host bits off and a Broadcast IP has all its host bits on, a Host IP can have any combination.

The range of a host is calculated adding one (integer) to the last byte in a Network IP and subtracting one to the last byte in a Broadcast IP.

For example the IP Address “192.54.3.21” is Class C, Type Host, its Network IP is “192.54.3.0” its Broadcast IP is “192.54.3.255”, its Range of host is from “192.54.3.1” to “192.54.1.254”.

Software (libraries, packages, tools):

The equipment used in the lab was:

-Laptop

-Pencil

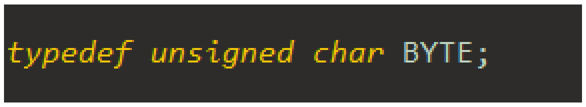
-Blank paper

The program was codified in C language using the text processor Sublime Text 3, compiled using the GNU Compiler.

Procedure:

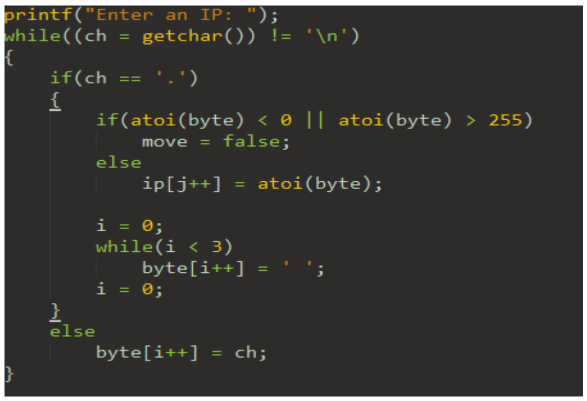
To do the program I followed the next steps:

First we have to get the data from the user in order to do this, we should save the IP entered by the user in such a way that we do not waste space in memory and also help us do the process more simple. Having that said the preferred option is to define a data type named BYTE, which is an unsigned char, since the standard C does not define a byte type.



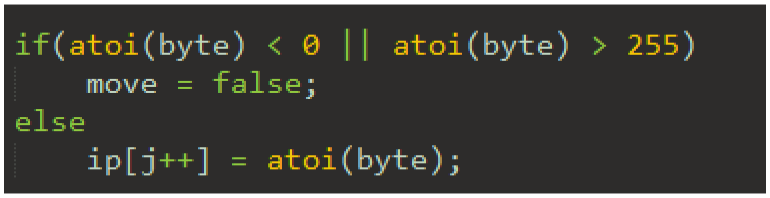
```
typedef unsigned char BYTE;
```

The we must receive the IP entered separated each part of the byte by a dot. We do so using a loop that reads the input character by character, saves the characters in a char array and transforms the char array into a "byte" using the function "atoi(char [])" and then saves it into a byte array named "ip" whenever it sees a ".", as follows:



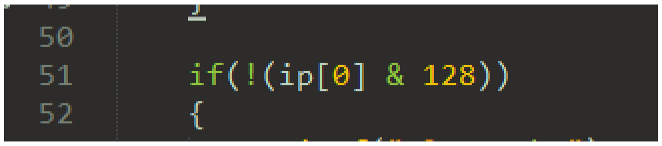
```
printf("Enter an IP: ");
while((ch = getchar()) != '\n')
{
    if(ch == '.')
    {
        if(atoi(byte) < 0 || atoi(byte) > 255)
            move = false;
        else
            ip[j++] = atoi(byte);

        i = 0;
        while(i < 3)
            byte[i++] = ' ';
        i = 0;
    }
    else
        byte[i++] = ch;
}
```

To validate the input we check whether the byte is less than 0 or greater than 255:



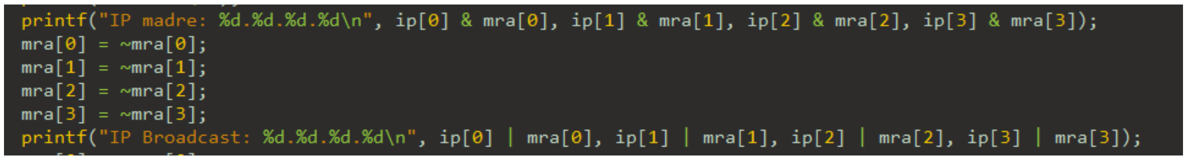
```
if(atoi(byte) < 0 || atoi(byte) > 255)
    move = false;
else
    ip[j++] = atoi(byte);
```

The next step is to check if the IP is Class A, B or C using the criteria gave in the literature part, for example we can determine if an IP is Class A by checking if it is in the range between 0 and 128, but a better way could be, to check if its first bit is on, as follows:



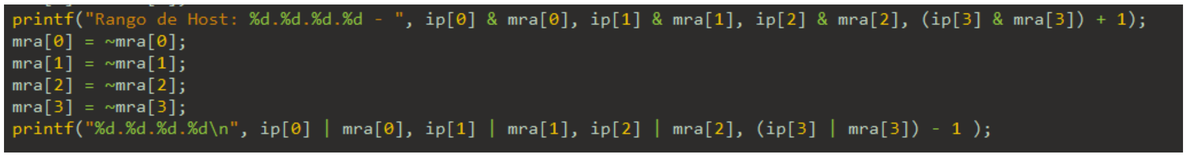
```

    if(!(ip[0] & 128))
    {
```

Once we have determined the Class we calculate the Network and Broadcast IP by making use of the respective Network Mask. The process for calculating a Network IP is to make an “&” between the bytes of the IP and the bytes of the Mask, to calculate the Broadcast we make a complement of the Mask and then an “|” with the IP:



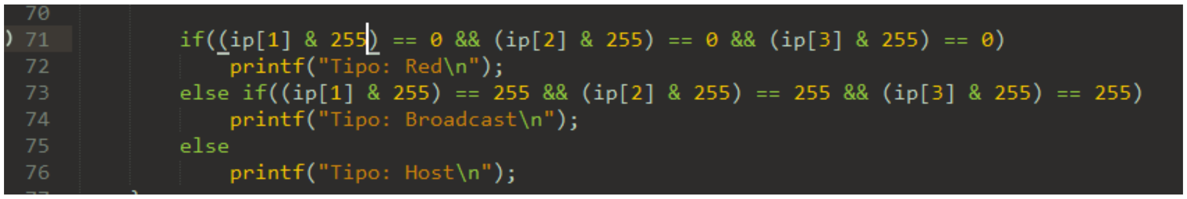
```
printf("IP madre: %d.%d.%d.%d\n", ip[0] & mra[0], ip[1] & mra[1], ip[2] & mra[2], ip[3] & mra[3]);
mra[0] = ~mra[0];
mra[1] = ~mra[1];
mra[2] = ~mra[2];
mra[3] = ~mra[3];
printf("IP Broadcast: %d.%d.%d.%d\n", ip[0] | mra[0], ip[1] | mra[1], ip[2] | mra[2], ip[3] | mra[3]);
```

To print the Range of Host we use the same process but adding one to the las byte of the Network IP and subtracting one from the Broadcast IP:



```
printf("Rango de Host: %d.%d.%d.%d - ", ip[0] & mra[0], ip[1] & mra[1], ip[2] & mra[2], (ip[3] & mra[3]) + 1);
mra[0] = ~mra[0];
mra[1] = ~mra[1];
mra[2] = ~mra[2];
mra[3] = ~mra[3];
printf("%d.%d.%d.%d\n", ip[0] | mra[0], ip[1] | mra[1], ip[2] | mra[2], (ip[3] | mra[3]) - 1 );
```

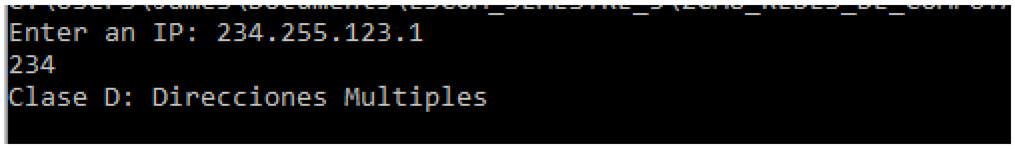
Finally, to check the Type we use the criteria we saw at the Literature Part:



```

    if((ip[1] & 255) == 0 && (ip[2] & 255) == 0 && (ip[3] & 255) == 0)
        printf("Tipo: Red\n");
    else if((ip[1] & 255) == 255 && (ip[2] & 255) == 255 && (ip[3] & 255) == 255)
        printf("Tipo: Broadcast\n");
    else
        printf("Tipo: Host\n");
```

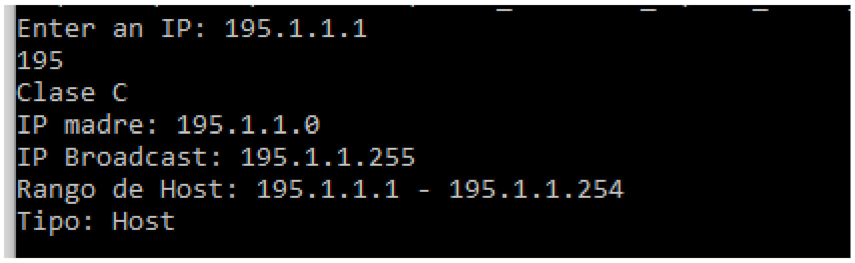
Results

The result of entering the IP “234.255.123.1”



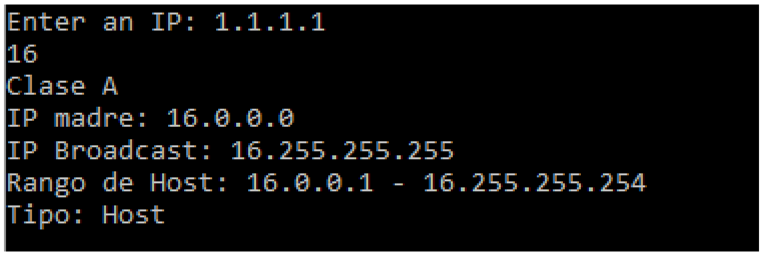
```
Enter an IP: 234.255.123.1
234
Clase D: Direcciones Multiples
```

The result of entering “195.1.1.1”



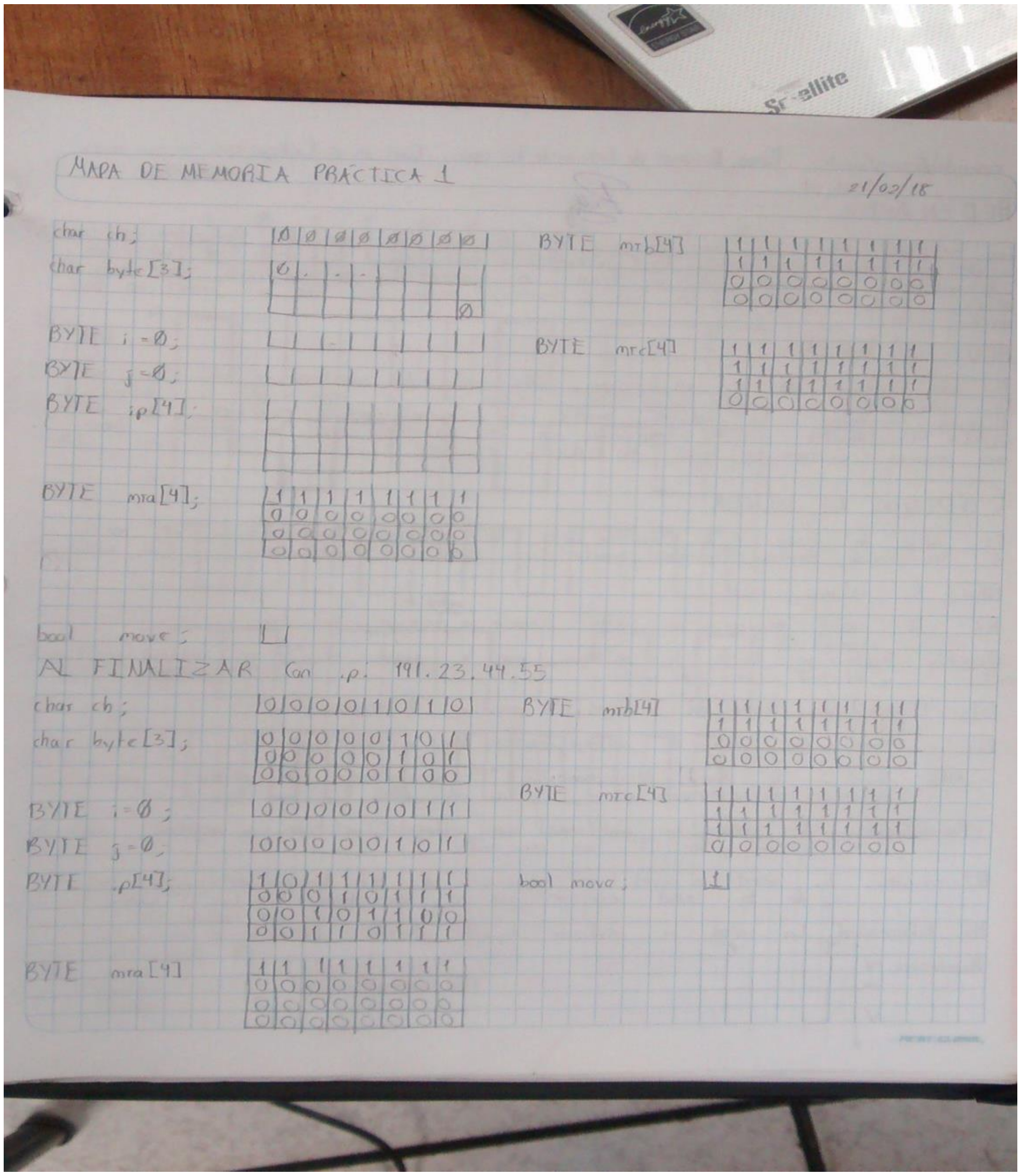
```
Enter an IP: 195.1.1.1
195
Clase C
IP madre: 195.1.1.0
IP Broadcast: 195.1.1.255
Rango de Host: 195.1.1.1 - 195.1.1.254
Tipo: Host
```

The result of “1.1.1.1”



```
Enter an IP: 1.1.1.1
16
Clase A
IP madre: 16.0.0.0
IP Broadcast: 16.255.255.255
Rango de Host: 16.0.0.1 - 16.255.255.254
Tipo: Host
```

A memory map showing the values of the actual variables used in the program before and after a process:



MAPA DE MEMORIA PRÁCTICA 1

21/02/18

char ch;

Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

char byte[3];

Ø	.	.	.				
							Ø

BYTE i = Ø;

BYTE j = Ø;

BYTE ip[4];

BYTE mra[4];

1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

BYTE mrb[4]

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

BYTE mrc[4]

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0

bool move;

AL FINALIZAR Con ip: 191.23.44.55

char ch;

0	0	0	0	1	0	1	0

char byte[3];

0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	1
0	0	0	0	0	1	0	0

BYTE i = Ø;

0	0	0	0	0	0	1	1

BYTE j = Ø;

0	0	0	0	0	1	0	1

BYTE ip[4];

1	0	1	1	1	1	1	1
0	0	0	1	0	1	1	1
0	0	1	0	1	1	0	0
0	0	1	1	0	1	1	1

BYTE mra[4]

1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

BYTE mrb[4]

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

BYTE mrc[4]

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0

bool move;

1

Discussion:

The results of the program are shown in console which may result not very visually pleasant for some users and that’s a drawback. However, the results are accurate and reliable, and the program is very portable and flexible as we can change the Network Mask at any time we want in order to calculate the characteristics of another type of IPs of Subnetworks.

Conclusions:

One thing I learned by doing this practice is to know how to use the bitwise operators C language provide, which operate on integer data at the bit level. It can be applied in real life to several programs that require to manage data at bit level, or to make programs more efficient since working at bit level is faster to compile and to understand by the processor.

The results that the program throws are always correct as long as the Mask and user’s input is correct, the program can be used with any IP.

The only defect I found in the program is that the calculus of the Broadcast and Network IP are made two times since we require to show them again for the Range of Host.

References:

K. N. King, C Programming A Modern Approach Second Edition. W. W. Norton, 2008

Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

typedef unsigned char BYTE;

int main(void)
{
    char ch;
    char byte[3];
    BYTE i = 0, j = 0;
    BYTE ip[4] = {0};
    bool move = true;
    BYTE mra[4] = {255, 0, 0, 0};
    BYTE mrb[4] = {255, 255, 0, 0};
    BYTE mrc[4] = {255, 255, 255, 0};

    for(;;)
    {
        i = 0;
        j = 0;
        printf("Enter an IP: ");
        while((ch = getchar()) != '\n')
        {
            if(ch == '.')
            {
                if(atoi(byte) < 0 || atoi(byte) > 255)
                    move = false;
                else
                    ip[j++] = atoi(byte);

                i = 0;
                while(i < 3)
                    byte[i++] = ' ';
                i = 0;
            }
            else
                byte[i++] = ch;
        }

        ip[j] = atoi(byte); //Guarda ip[3]

        if(move == true)
            break;
        else
            move = true;

        printf("Error: Entrada no valida\n");
    }

    if(!(ip[0] & 128))
    {
        printf("Clase A\n");
        printf("IP madre: %d.%d.%d.%d\n", ip[0] & mra[0], ip[1] & mra[1], ip[2] & mra[2], ip[3] & mra[3]);
        mra[0] = ~mra[0];
        mra[1] = ~mra[1];
        mra[2] = ~mra[2];
        mra[3] = ~mra[3];
        printf("IP Broadcast: %d.%d.%d.%d\n", ip[0] | mra[0], ip[1] | mra[1], ip[2] | mra[2], ip[3] | mra[3]);
        mra[0] = ~mra[0];
        mra[1] = ~mra[1];
        mra[2] = ~mra[2];
        mra[3] = ~mra[3];
        printf("Rango de Host: %d.%d.%d.%d - ", ip[0] & mra[0], ip[1] & mra[1], ip[2] & mra[2], (ip[3] & mra[3]) + 1);
        mra[0] = ~mra[0];
        mra[1] = ~mra[1];
        mra[2] = ~mra[2];
        mra[3] = ~mra[3];
        printf("%d.%d.%d.%d\n", ip[0] | mra[0], ip[1] | mra[1], ip[2] | mra[2], (ip[3] | mra[3]) - 1 );

        if((ip[1] & 255) == 0 && (ip[2] & 255) == 0 && (ip[3] & 255) == 0)
            printf("Tipo: Red\n");
        else if((ip[1] & 255) == 255 && (ip[2] & 255) == 255 && (ip[3] & 255) == 255)
            printf("Tipo: Broadcast\n");
        else
            printf("Tipo: Host\n");
    }
    else if(!(ip[0] & 64))
    {
        printf("Clase B\n");
        printf("IP madre: %d.%d.%d.%d\n", ip[0] & mrb[0], ip[1] & mrb[1], ip[2] & mrb[2], ip[3] & mrb[3]);
        mrb[0] = ~mrb[0];
        mrb[1] = ~mrb[1];
        mrb[2] = ~mrb[2];
        mrb[3] = ~mrb[3];
        printf("IP Broadcast: %d.%d.%d.%d\n", ip[0] | mrb[0], ip[1] | mrb[1], ip[2] | mrb[2], ip[3] | mrb[3]);
        mrb[0] = ~mrb[0];
        mrb[1] = ~mrb[1];
        mrb[2] = ~mrb[2];
        mrb[3] = ~mrb[3];
        printf("Rango de Host: %d.%d.%d.%d - ", ip[0] & mrb[0], ip[1] & mrb[1], ip[2] & mrb[2], (ip[3] & mrb[3]) + 1);
        mrb[0] = ~mrb[0];
        mrb[1] = ~mrb[1];
        mrb[2] = ~mrb[2];
        mrb[3] = ~mrb[3];
        printf("%d.%d.%d.%d\n", ip[0] | mrb[0], ip[1] | mrb[1], ip[2] | mrb[2], (ip[3] | mrb[3]) - 1);

        if((ip[2] & 255) == 0 && (ip[3] & 255) == 0)
            printf("Tipo: Red\n");
            else if((ip[2] & 255) == 255 && (ip[3] & 255) == 255)
                printf("Tipo: Broadcast\n");
            else
                printf("Tipo: Host\n");
        }
        else if(!(ip[0] & 32))
        {
            printf("Clase C\n");
            printf("IP madre: %d.%d.%d.%d\n", ip[0] & mrc[0], ip[1] & mrc[1], ip[2] & mrc[2], ip[3] & mrc[3]);
            mrc[0] = ~mrc[0];
            mrc[1] = ~mrc[1];
            mrc[2] = ~mrc[2];
            mrc[3] = ~mrc[3];
            printf("IP Broadcast: %d.%d.%d.%d\n", ip[0] | mrc[0], ip[1] | mrc[1], ip[2] | mrc[2], ip[3] | mrc[3]);
            mrc[0] = ~mrc[0];
            mrc[1] = ~mrc[1];
            mrc[2] = ~mrc[2];
            mrc[3] = ~mrc[3];
            printf("Rango de Host: %d.%d.%d.%d - ", ip[0] & mrc[0], ip[1] & mrc[1], ip[2] & mrc[2], (ip[3] & mrc[3]) + 1);
            mrc[0] = ~mrc[0];
            mrc[1] = ~mrc[1];
            mrc[2] = ~mrc[2];
            mrc[3] = ~mrc[3];
            printf("%d.%d.%d.%d\n", ip[0] | mrc[0], ip[1] | mrc[1], ip[2] | mrc[2], (ip[3] | mrc[3]) - 1);

            if((ip[3] & 255) == 0)
                printf("Tipo: Red\n");
            else if((ip[3] & 255) == 255)
                printf("Tipo: Broadcast\n");
            else
                printf("Tipo: Host\n");
        }
        else if(!(ip[0] & 16))
        {
            printf("Clase D: Direcciones Multiples\n");
        }
        else
        {
            printf("Clase E: Experimentacion\n");
        }

        return 0;
    }
```