МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЕТ

по лабораторной работе № 11
по дисциплине «Программирование»
Тема: Битовые поля в структурах

Студент гр. 3312 _____ Мохно Даниил.

Преподаватель _____ Аббас Саддам

Санкт-Петербург
2024

Цель работы.

Целью работы является изучение битовых полей в структурах в языке Си и получение практических навыков в программировании на этом языке.

Задание (вариант 1)

Числовой адрес компьютера в глобальной сети Интернет (ip-адрес) версии 4 состоит из 4-х чисел от 0 до 255, разделенных точками (например, 123.45.67.89). Для записи каждого числа используется 1 байт (октет). Значения битов первого октета определяют т. н. «класс сети».

0xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx — класс A

10xxxxxx xxxxxxxx xxxxxxxx xxxxxxxx — класс B

110xxxxx xxxxxxxx xxxxxxxx xxxxxxxx — класс C

(x — произвольное значение бита — 0 или 1)

Разработать алгоритм и реализовать функции преобразования произвольного адреса IPv4 класса A в адрес класса B и наоборот с использованием битовых полей в структурах и битовых операций.

Адреса вводятся с клавиатуры.

Постановка задачи и описание решения

Создадим объединение ip адреса IPv4, оно будет содержать массив address из четырёх элементов 8-и битного целочисленного беззнакового типа, и структуру, содержащую неименованное поле на 6 бит и целочисленное поле class на 2 бита, которое будет отвечать за последние два бита первого октета.

В главной функции выделяем место под объединение, которое будет хранить ip адрес, вызываем функцию получения ip адреса, которой параметром передаём указатель на объединение и получаем от него флаг удавшегося получения: 1–получено 0–не получено. Если ip не получен, то скорее всего пользователь ввёл адрес, не проходящий по формату, и мы выводим сообщение об ошибке. В противном случае, вызываем функцию вывода ip адреса, ей параметром передаём тот же указатель, затем вызываем функцию смены класса

ip адреса, ей так же передаём указатель на объединение, и вызываем функцию вывода ip адреса после чего освобождаем место из-под объединения.

В функции получения ip адреса получаем через форматированный ввод октеты ip адреса, которые сохраняем в ячейках массива временного хранения. Затем ставим флаг полученного адреса 1, и запускаем цикл на 4 итерации, в котором проверяем i-й элемент массива, чтобы он был меньше 256. Если условие выполняется, копируем i-й элемент временного массива в i-ю ячейку массива нашего объединения. Если не выполняется, присваиваем флагу 0. После завершения цикла возвещаем флаг

В функции вывода ip адреса, мы выводим все октеты из массива в объединении, а затем циклом проходимся по массиву вызывая в каждой итерации функцию вывода двоичного представления числа, предавая ей число из массива.

В функции вывода двоичного представления числа мы запускаем цикл на 8 итераций, ставя итератор i равным 7, и пока он больше или равен 0. В цикле выводим число, полученное путём побитового сдвига в право на i позиций и побитового «и» с единицей. То есть мы сдвинем число до текущего бита, а все остальные биты обнулим. Таким образом получаем текущий бит.

В функции смены класса ip в поле class объединения запишем результат операции исключающего или между текущим значением поля class (последние 2 бита адреса) и 0b10 (битовым представлением 2-и). За тем проверяем является ли второй бит 1-ей путем операции битового «и» между class и 0b10, если да, то записываем полученное в сравнении значение в class.

Описание переменных

Функция – int main():

№	Имя переменной	Тип	Назначение
1	ip	struct IPv4	Указатель на объединение с ip адресом
2	complete	int	Флаг, указывающий на то, что ip был получен

Функция функция получения ip адреса – void get_ip(IPv4 *ip):

№	Имя переменной	Тип	Назначение
1	ip	struct IPv4	Указатель на объединение с ip адресом
2	address	int	Массив для временного хранения полученного от пользователя адреса
3	complete	int	Флаг, указывающий на то, что ip был получен
4	i	int	Итератор

Функция вывода одного байта в битовом формате – void short_to_bin(IPv4 *ip):

№	Имя переменной	Тип	Назначение
1	address	u_int_8t	Однобайтное целое беззнаковое число, содержащее октет ip адреса
2	i	int	Итератор

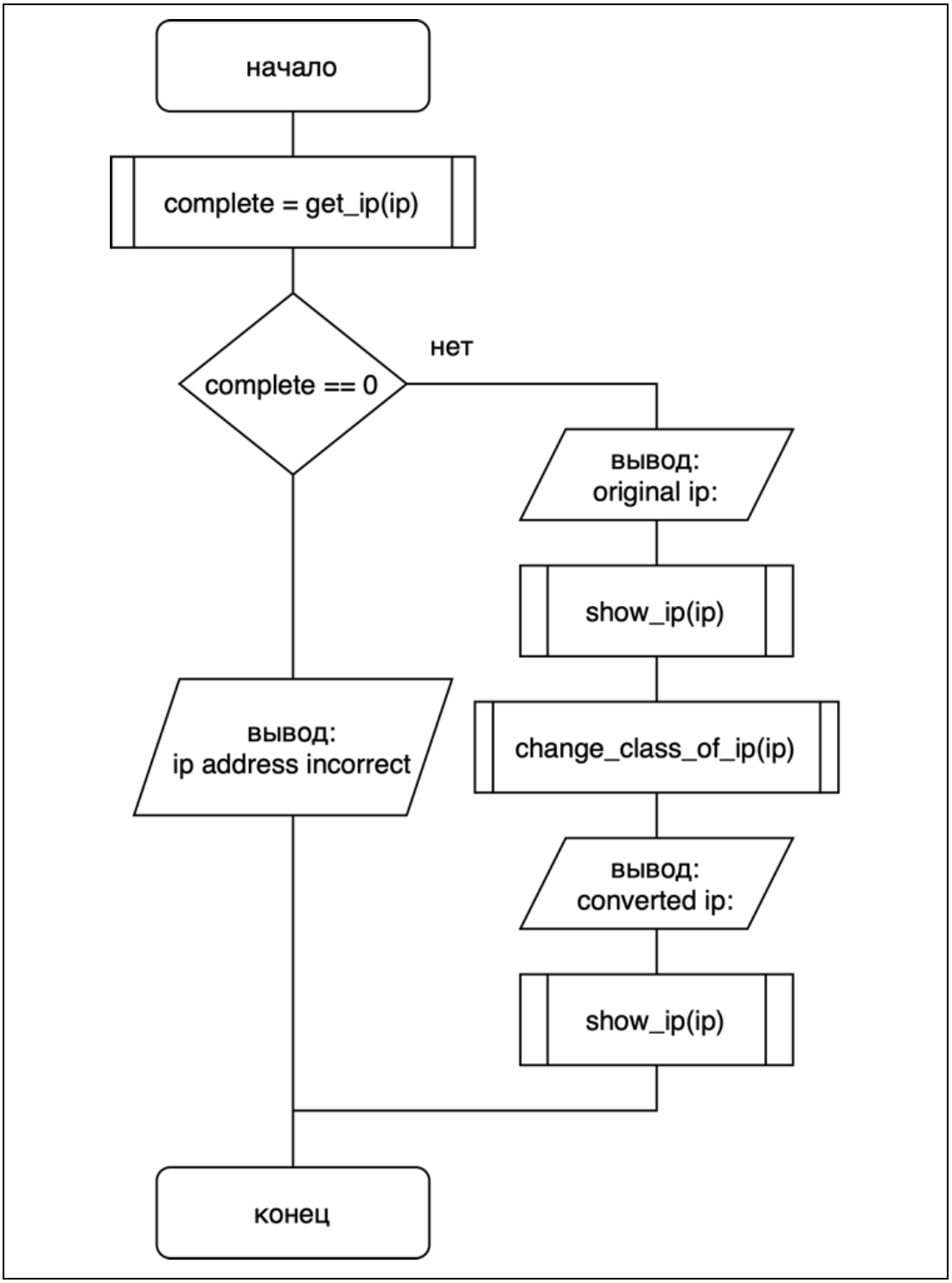
Функция замены двух старших битов первого октета ip адреса – void change_class_of_ip(IPv4 *ip)

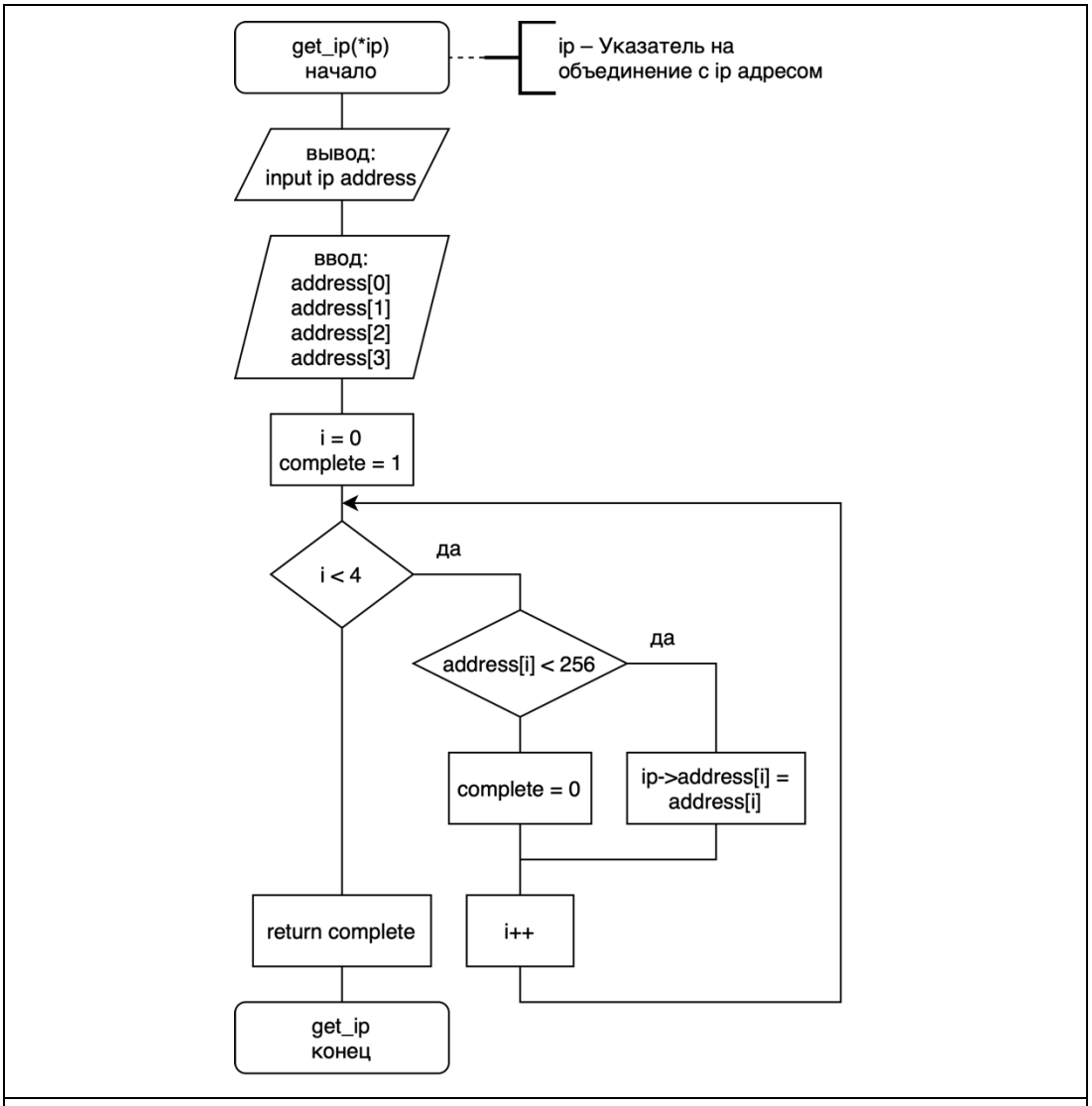
№	Имя переменной	Тип	Назначение
1	ip	struct IPv4	Указатель на объединение с ip адресом

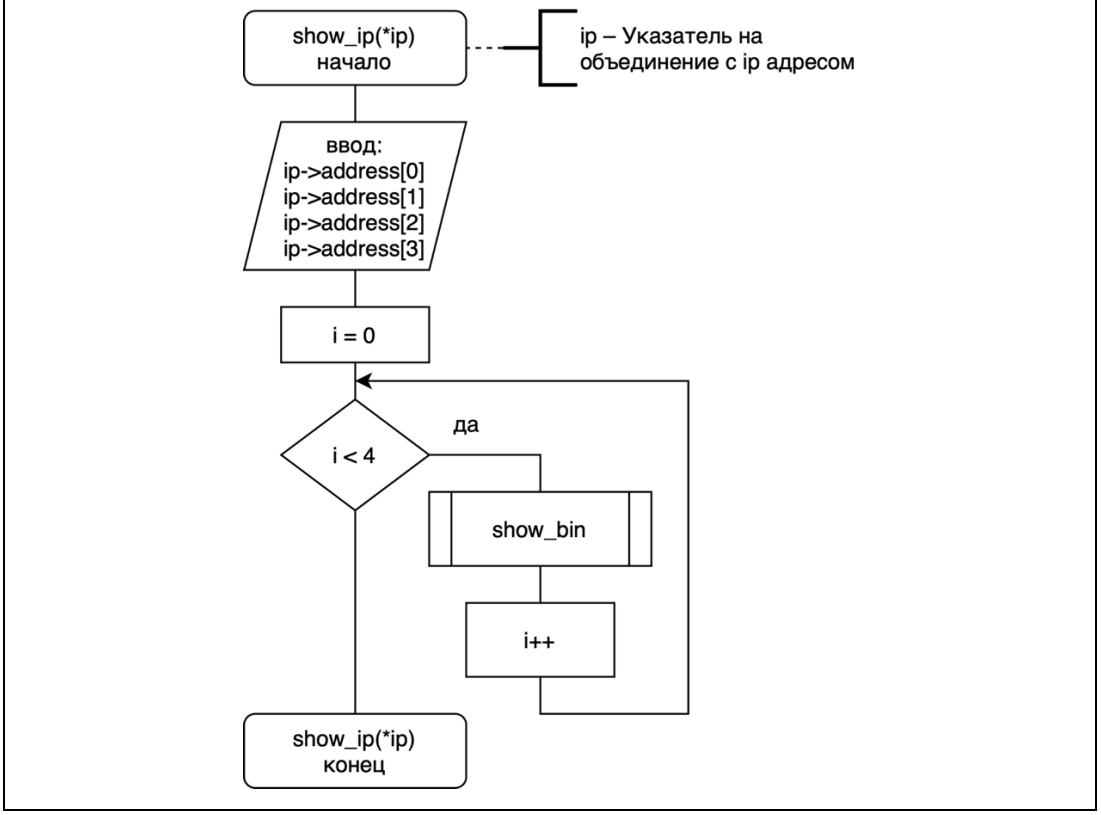
Функция вывода ip адреса и его двоичного битового представления – void show_ip(IPv4 *ip)

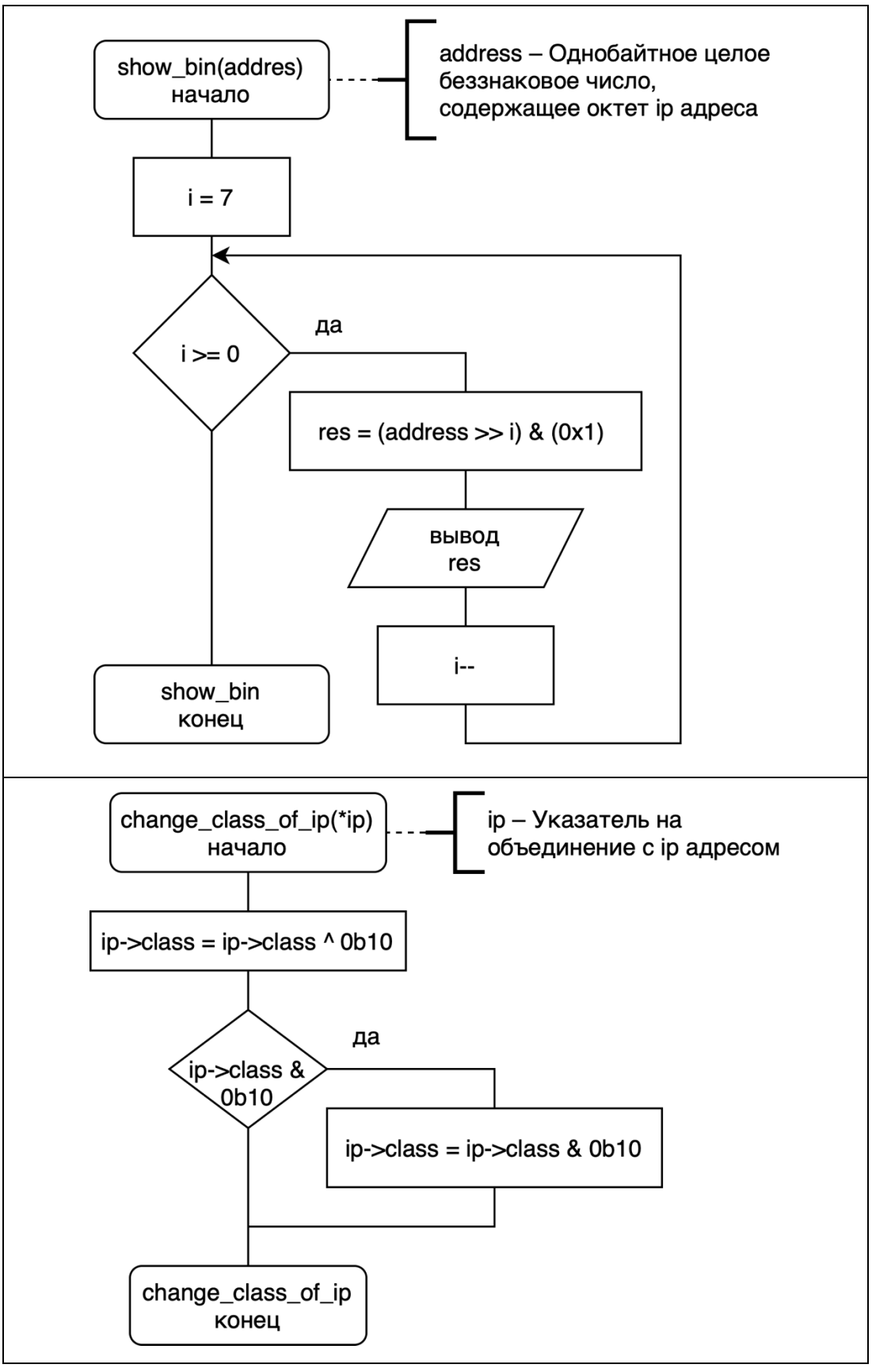
№	Имя переменной	Тип	Назначение
1	ip	struct IPv4	Указатель на объединение с ip адресом
2	i	int	Итератор

Схема алгоритма




get_ip(*ip)
начало
ip – Указатель на
объединение с ip адресом
вывод:
input ip address
ввод:
address[0]
address[1]
address[2]
address[3]
i = 0
complete = 1
i < 4
да
address[i] < 256
да
complete = 0
ip->address[i] =
address[i]
return complete
i++
get_ip
конец


show_ip(*ip)
начало
ip – Указатель на
объединение с ip адресом
ввод:
ip->address[0]
ip->address[1]
ip->address[2]
ip->address[3]
i = 0
i < 4
да
show_bin
i++
show_ip(*ip)
конец


show_bin(addres)
начало
address – Однобайтное целое
беззнаковое число,
содержащее октет ip адреса
i = 7
i >= 0
да
res = (address >> i) & (0x1)
вывод
res
i--
show_bin
конец
change_class_of_ip(*ip)
начало
ip – Указатель на
объединение с ip адресом
ip->class = ip->class ^ 0b10
ip->class &
0b10
да
ip->class = ip->class & 0b10
change_class_of_ip
конец

Контрольные примеры

№	Входные данные	Выходные данные
1	10.100.100.100	original ip: 10.100.100.100 00001010.01100100.01100100.01100100 converted ip: 138.100.100.100 10001010.01100100.01100100.01100100
2	123.45.67.89	original ip: 123.45.67.89 01111011.00101101.01000011.01011001 converted ip: 187.45.67.89 10111011.00101101.01000011.01011001
3	172.31.255.254	original ip: 172.31.255.254 10101100.00011111.11111111.11111110 converted ip: 44.31.255.254 00101100.00011111.11111111.11111110
4	256.88.90.23	Address incorrect

Текст программы

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

typedef union IPv4
{
    u_int8_t address[4];
    struct
    {
        int : 6;
        int class: 2;
    };
} IPv4;

void get_ip(IPv4 *ip);

void show_ip(IPv4 *ip);

void show_bin(u_int8_t address);

void change_class_of_ip(IPv4 *ip);

int main()
{
    IPv4 *ip;
    int complete;
```

```
    ip = malloc(sizeof(IPv4));
    complete = get_ip(ip);
    if (!complete) printf("ip address incorrect");
    else
    {
        puts("original ip:");
        show_ip(ip);
        change_class_of_ip(ip);
        puts("converted ip:");
        show_ip(ip);
    }
    free(ip);
    return 0;
}

int get_ip(IPv4 *ip)
{
    int i, complete;
    int address[4];
    printf("input ip address\n");
    scanf("%d.%d.%d.%d", &(address[0]), &(address[1]), &(address[2]), &(address[3]));
    complete = 1;
    for (i = 0; i < 4; i++)
        if (address[i] < 256)
            ip->address[i] = address[i];
        else complete = 0;
    return complete;
}

void show_ip(IPv4 *ip)
{
    int i;
    printf("%hhu.%hhu.%hhu.%hhu\n", ip->address[0], ip->address[1], ip->address[2], ip-
>address[3]);
    for (i = 0; i < 4; i++)
    {
        show_bin(ip->address[i]);
        putchar('.');
    }
    putchar(8);
    putchar('\n');
}

void show_bin(u_int8_t address)
{
    int i;
    for (i = 7; i >= 0; i--) printf("%d", (address >> i) & (0x1));
}

void change_class_of_ip(IPv4 *ip)
{
    ip->class = ip->class ^ 0b10;
    if (ip->class & 0b10) ip->class = ip->class & 0b10;
}
```

Примеры выполнения программы

Пример 1:

```
input ip address
10.100.100.100
original ip:
10.100.100.100
00001010.01100100.01100100.01100100
converted ip:
138.100.100.100
10001010.01100100.01100100.01100100
```

Пример 2:

```
input ip address
123.45.67.89
original ip:
123.45.67.89
01111011.00101101.01000011.01011001
converted ip:
187.45.67.89
10111011.00101101.01000011.01011001
```

Пример 3:

```
input ip address
172.31.255.254
original ip:
172.31.255.254
10101100.00011111.11111111.11111110
converted ip:
44.31.255.254
00101100.00011111.11111111.11111110
```

Пример 4:

```
input ip address
256.88.90.23
Address incorrect
```

Выводы.

В результате выполнения работы были изучены битовые поля в структурах в языке Си и получены практические навыки в программировании на этом языке.