Язык программирования C++

Отчет

14.09.2020

Практическое занятие 2
Типы данных, операции

Задача №1

1. Условия задачи

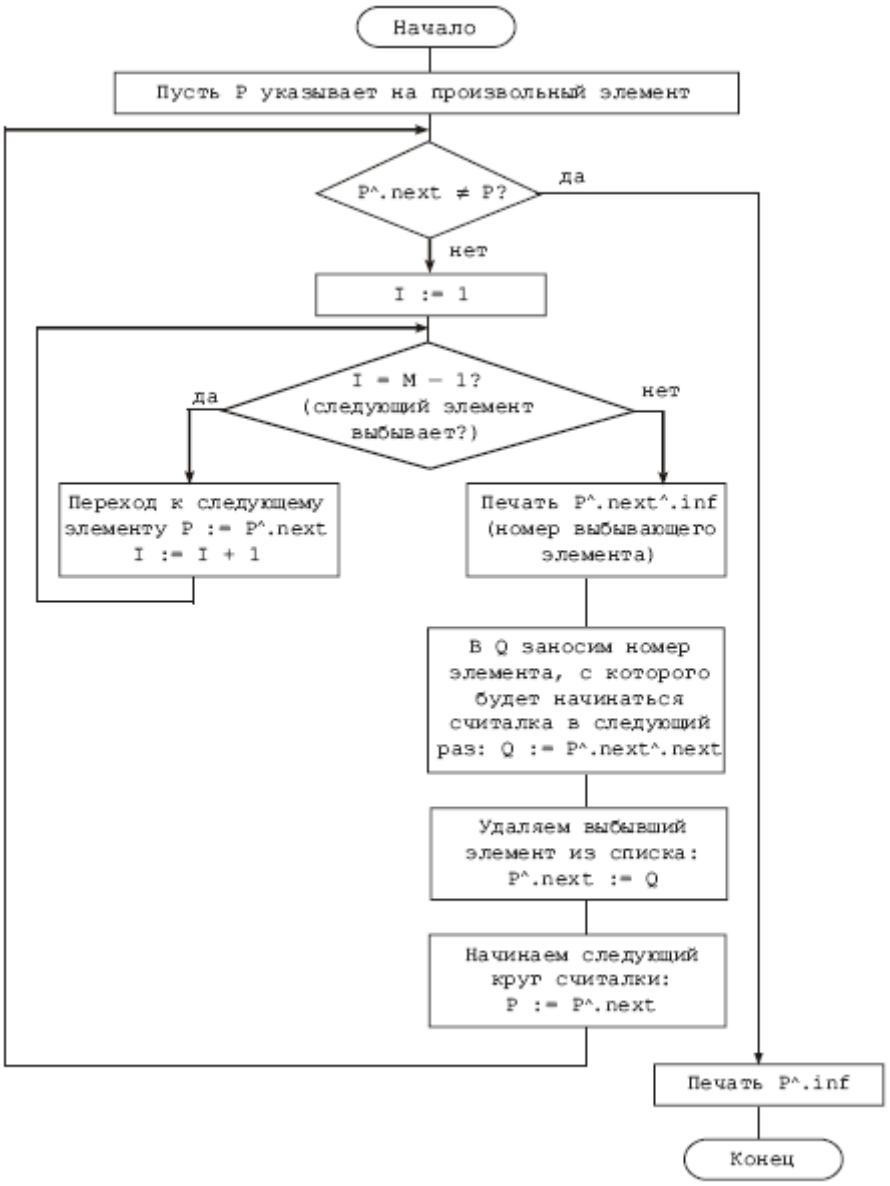
Пример №22 из лекции. Вам дана маска подсети и n IP-адресов. Ваша задача определить сколько различных адресов сети получится.

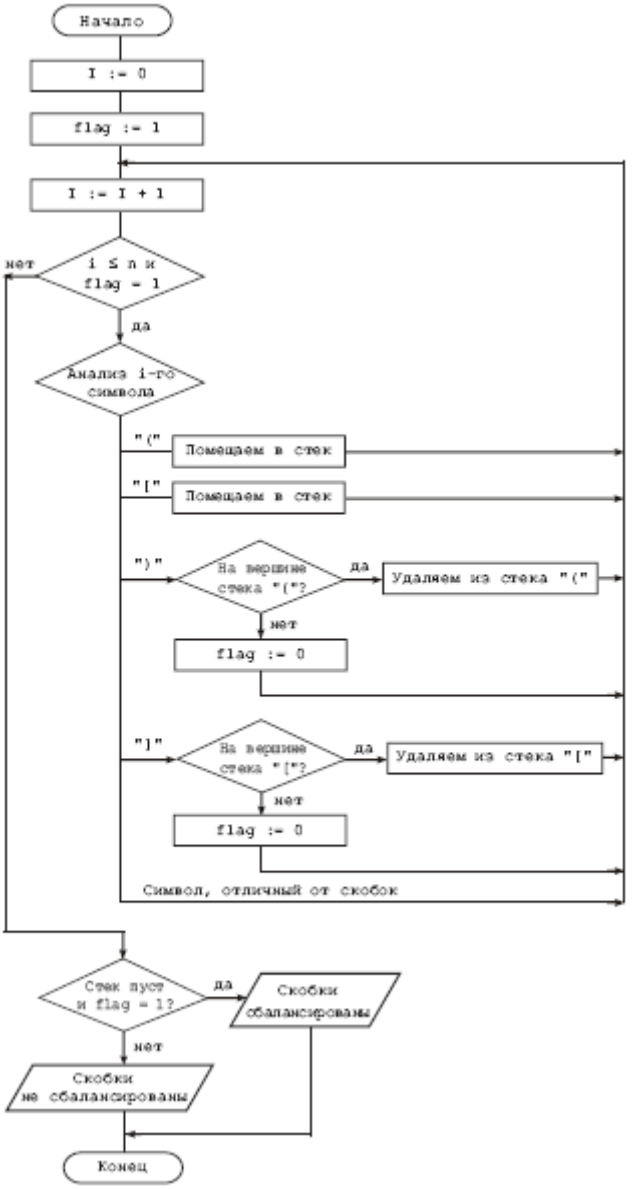
2. Алгоритм. Блок-схема

Поразрядная конъюнкция — применение к соответствующим битам операции логического И. Другими словами, если соответствующие биты равны 1, результирующий двоичный разряд будет равен 1, иначе 0.

Применение поразрядной конъюнкции к маске подсети и IP-адресу выглядит так:

```
11111111.11111111.11111111.00000000 (255.255.255.0)
&
11000000.10101000.00000000.00000001 (192.168.0.1)
-----------------------------------
11000000.10101000.00000000.00000000 (192.168.0.0) <--- Адрес сети
```


Начало
Пусть P указывает на произвольный элемент
P^.next ≠ P?
да
нет
I := 1
I = M – 1?
(следующий элемент
выбывает?)
да
нет
Переход к следующему
элементу P := P^.next
I := I + 1
Печать P^.next^.inf
(номер выбывающего
элемента)
В Q заносим номер
элемента, с которого
будет начинаться
считалка в следующий
раз: Q := P^.next^.next
Удаляем выбывший
элемент из списка:
P^.next := Q
Начинаем следующий
круг считалки:
P := P^.next
Печать P^.inf
Конец


Начало
I := 0
flag := 1
I := I + 1
нет
i ≤ n и
flag = 1
да
Анализ i-го
символа
"("
Помещаем в стек
"["
Помещаем в стек
")"
На вершине
стека "("?
да
Удаляем из стека "("
нет
flag := 0
"]"
На вершине
стека "["?
да
Удаляем из стека "["
нет
flag := 0
Символ, отличный от скобок
Стек пуст
и flag = 1?
да
Скобки
сбалансированы
нет
Скобки
не сбалансированы
Конец

3. Исходный код программы

```cpp
#include <iostream>
#include <vector>
#include <set>

using namespace std;

int address(int p1, int p2, int p3, int p4) {
        int data = p1;
        data <<= 8;
        data |= p2;
        data <<= 8;
        data |= p3;
        data <<= 8;
        data |= p4;
        return data;
}

int read_ad() {
        int m1, m2, m3, m4;
        char c;
        cin >> m1 >> c >> m2 >> c >> m3 >> c >> m4;
        return address(m1, m2, m3, m4);
}

set <int> s;

int main() {
        int mask = read_ad();
        int n;
        cin >> n;
        for (int i = 0; i < n; i++) {
                int x = read_ad();
                s.insert(mask & x);
        }
        cout << s.size();
        return 0;
}
```

4. Формат входных и выходных данных

Формат входных данных

В первой строке вам дана маска сети. Во второй строке дано числа n — количество IP-адресов ($1 \leqslant n \leqslant 1000$). Следующие n строк содержат IP-адреса.

Формат выходных данных

Выведите количество различных адресов сети.

5. Результат работы программы

Стандартный ввод
255.255.255.0 3 192.168.0.1 192.168.0.28 192.168.1.1
Стандартный вывод
2

6. Выводы и комментарии к решению задачи

p130_seven_segment_counter.ino

```
#define FIRST_SEGMENT_PIN   2
#define SEGMENT_COUNT       7

// префикс «0b» означает, что целое число за ним записано в
// в двоичном коде. Единицами мы обозначим номера сегментов
// индикатора, которые должны быть включены для отображения
// арабской цифры. Всего цифр 10, поэтому в массиве 10 чисел.
// Нам достаточно всего байта (англ. byte, 8 бит) для хранения
// комбинации сегментов для каждой из цифр.
byte numberSegments[10] = {
  0b00111111, 0b00001010, 0b01011101, 0b01011110, 0b01101010,
  0b01110110, 0b01110111, 0b00011010, 0b01111111, 0b01111110,
};

void setup()
{
  for (int i = 0; i < SEGMENT_COUNT; ++i)
    pinMode(i + FIRST_SEGMENT_PIN, OUTPUT);
}

void loop()
{
  // определяем число, которое собираемся отображать. Пусть им
  // будет номер текущей секунды, зацикленный на десятке
  int number = (millis() / 1000) % 10;
  // получаем код, в котором зашифрована арабская цифра
  int mask = numberSegments[number];
  // для каждого из 7 сегментов индикатора...
  for (int i = 0; i < SEGMENT_COUNT; ++i) {
    // ...определяем: должен ли он быть включён. Для этого
    // считываем бит (англ. read bit), соответствующий текущему
    // сегменту «i». Истина — он установлен (1), ложь — нет (0)
    boolean enableSegment = bitRead(mask, i);
    // включаем/выключаем сегмент на основе полученного значения
    digitalWrite(i + FIRST_SEGMENT_PIN, enableSegment);
  }
}
```

Пояснения к коду

- Мы создали массив типа `byte`: каждый его элемент это 1 байт, 8 бит, может принимать значения от 0 до 255.
- Символы арабских цифр закодированы состоянием пинов, которые соединены с выводами соответствующих сегментов: 0, если сегмент должен быть выключен, и 1, если включен.
- В переменную `mask` мы помещаем тот элемент массива `numberSegments`, который соответствует текущей секунде, вычисленной в предыдущей инструкции.
- В цикле `for` мы пробегаем по всем сегментам, извлекая с помощью встроенной функции `bitRead` нужное состояние для текущего пина, в которое его и приводим с помощью `digitalWrite` и переменной `enableSegment`
- `bitRead(x, n)` возвращает `boolean` значение: **n**-ный бит *справа* в байте **x**

Задача №2

...

Задача №3

...

Выводы по лабораторной работе 3

1. Решены 5 задач по теме «Линейные и разветвляющиеся алгоритмы»:
1) о маске подсети,
2) о ...
3) на вычисление суммы массива,
4) ...
5) ...

2. Для решения 5 задачи не достаточно знаний
о выводе в *бинарный* файл.

СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Бланшет, Ж. Qt 4: программирование GUI на C++. Пер. с англ. 2-е изд., доп. / Ж.Бланшет, М.Саммерфилд. – М.: Кудиц-пресс, 2008. – 736 с.
2. Прикладное программирование. – Режим доступа: https://ifizmat.16mb.com/application, свободный. Загл. с экрана. – Дата обращения: 14.04.2017