```
//Write a C++ program to find the Armstrong number for a given range of number.
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
   int start,end,i,j;
   cout<<"Enter the starting number = ";
   cin>>start;//5
   cout<<"Enter the ending number = ";
   cin>>end;//15
   for(i=start;i<=end;i++)
   {
      int num = i;
      int temp = i;
      int digit=0,sum=0;
      while(num!=0)
      {
         num = num/10;
         digit++;
      }
      for(j=1;j<=digit;j++)
      {
         int rem = temp%10;
         int power = pow(rem,digit);
         sum = sum + power;
```

```
        temp = temp/10;
      }
      if(sum==i)
      {
        cout<<i<<" ";
      }
    }
    return 0;
}
```

Program ka Logic

1. **Input lena**:
    a. Program user se do numbers mangta hai: `start` aur `end`. Yeh range define karte hain jismein Armstrong numbers dhundne hain.
    b. Example: Agar `start = 5` aur `end = 15` hai, toh program 5 se 15 tak har number check karega.
2. **Armstrong number kya hota hai?**:
    a. Ek number Armstrong number hota hai agar uske digits ke cubes (ya digits ki power, jahan power utni hi hoti hai jitne digits hain) ka sum usi number ke barabar ho.
    b. Example: 153 ek Armstrong number hai kyunki:
        i. Digits: 1, 5, 3 (total 3 digits)
        ii. Sum: (1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153)
        iii. Yeh sum original number ke barabar hai, toh 153 Armstrong hai.
3. **Kaise kaam karta hai?**:
    a. **Outer loop**: `start` se `end` tak har number `i` ke liye check karta hai.

b. **Digits count karna**:
    i. Har number ke digits count karne ke liye ek `while` loop chalega. Number ko 10 se divide karte jao jab tak number 0 na ho jaye. Har division pe `digit` variable increment hota hai.
c. **Sum calculate karna**:
    i. Ek `for` loop chalega jo har digit ko nikalta hai (`temp%10`), uski power (digit count ke barabar) calculate karta hai (`pow(rem, digit)`), aur sum mein add karta hai.
d. **Check karna**:
    i. Agar calculated `sum` original number `i` ke barabar hai, toh woh number print ho jayega.

Dry Run

Ab ek example ke saath dry run karte hain. Maan le `start = 5` aur `end = 15`.

Input:

- `start = 5`
- `end = 15`

Outer Loop:

- Loop chalega `i = 5` se `i = 15` tak.

i = 5:

1. `num = i = 5`, `temp = i = 5`, `digit = 0`, `sum = 0`
2. **Digits count karna**:
    a. `while(num != 0)`:
        i. `num = 5 / 10 = 0` (integer division)
        ii. `digit++` → `digit = 1`
        iii. Loop khatam kyunki `num = 0`

b. Toh `digit = 1` (5 mein 1 digit hai)

3. **Sum calculate karna**:
   a. `for(j = 1; j <= digit; j++)` → sirf ek baar chalega kyunki `digit = 1`
      i. `rem = temp % 10 = 5 % 10 = 5`
      ii. `power = pow(rem, digit) = pow(5, 1) = 5`
      iii. `sum = sum + power = 0 + 5 = 5`
      iv. `temp = temp / 10 = 5 / 10 = 0`
   b. Loop khatam
4. **Check**:
   a. `sum = 5, i = 5`
   b. `sum == i` → `5 == 5` (true)
   c. Toh `cout << i << " ";` → **5 print hoga**

i = 6:

1. `num = 6, temp = 6, digit = 0, sum = 0`
2. **Digits count**:
   a. `num = 6 / 10 = 0`
   b. `digit++` → `digit = 1`
3. **Sum calculate**:
   a. `rem = 6 % 10 = 6`
   b. `power = pow(6, 1) = 6`
   c. `sum = 0 + 6 = 6`
   d. `temp = 6 / 10 = 0`
4. **Check**:
   a. `sum = 6, i = 6`
   b. `sum == i` → true
   c. 6 print hoga

i = 7:

1. `num = 7, temp = 7, digit = 0, sum = 0`
2. **Digits count**:

    a. `num = 7 / 10 = 0`
    b. `digit++ → digit = 1`
3. **Sum calculate**:
    a. `rem = 7 % 10 = 7`
    b. `power = pow(7, 1) = 7`
    c. `sum = 0 + 7 = 7`
    d. `temp = 7 / 10 = 0`
4. **Check**:
    a. `sum = 7, i = 7`
    b. `sum == i` → true
    c. 7 print hoga

i = 8:

1. `num = 8, temp = 8, digit = 0, sum = 0`
2. **Digits count**:
    a. `num = 8 / 10 = 0`
    b. `digit++ → digit = 1`
3. **Sum calculate**:
    a. `rem = 8 % 10 = 8`
    b. `power = pow(8, 1) = 8`
    c. `sum = 0 + 8 = 8`
    d. `temp = 8 / 10 = 0`
4. **Check**:
    a. `sum = 8, i = 8`
    b. `sum == i` → true
    c. 8 print hoga

i = 9:

- Same process, result: **9 print hoga**

i = 10:

1. `num = 10, temp = 10, digit = 0, sum = 0`
2. **Digits count**:
    a. `num = 10 / 10 = 1`
    b. `digit++ → digit = 1`
    c. `num = 1 / 10 = 0`
    d. `digit++ → digit = 2`
3. **Sum calculate**:
    a. `for(j = 1; j <= 2; j++)`:
        i. First iteration:
            1. `rem = 10 % 10 = 0`
            2. `power = pow(0, 2) = 0`
            3. `sum = 0 + 0 = 0`
            4. `temp = 10 / 10 = 1`
        ii. Second iteration:
            1. `rem = 1 % 10 = 1`
            2. `power = pow(1, 2) = 1`
            3. `sum = 0 + 1 = 1`
            4. `temp = 1 / 10 = 0`
4. **Check**:
    a. `sum = 1, i = 10`
    b. `sum != i` → false
    c. 10 print nahi hoga

i = 11 to 15:

- Similarly, yeh numbers check honge, lekin inme se koi bhi Armstrong number nahi hai kyunki:
    - 2-digit numbers ke liye, har digit ki square ka sum original number ke barabar nahi aata.
    - Example for 11:
        - Digits: 1, 1

- Sum: (1^2 + 1^2 = 1 + 1 = 2 \neq 11)

Output:

- 5 se 15 ke beech Armstrong numbers hain: **5, 6, 7, 8, 9**
- Toh output hoga: 5 6 7 8 9

Kuch Important Points:

- Yeh program single-digit numbers (1-9) ke liye hamesha Armstrong number dega kyunki (n^1 = n).
- 2-digit ya zyada digits ke numbers ke liye, sum match karna mushkil hota hai, isliye is range mein sirf single-digit numbers hi Armstrong hain.
- `pow` function floating-point values deta hai, lekin yahan integer context mein use hota hai, toh result sahi aata hai.