



LABORATORY 4

April 9, 2021

Abstract

We learned to use multiple functions in order to perform calculations accurately and to format the codes

Jordan Xu

Ece 1310 C for Engineers

- Given single-character codes for the colored bands that mark a resistor, compute its resistance. The color codes are as follows:

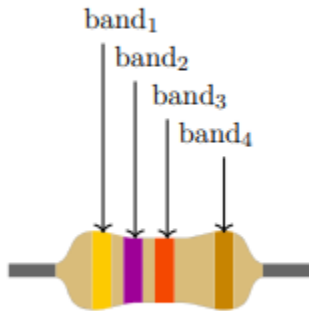


Figure 1: Resistor Color Bands

Color	Code	Character Code
Black	0	'B'
Brown	1	'N'
Red	2	'R'
Orange	3	'O'
Yellow	4	'Y'
Green	5	'G'
Blue	6	'E'
Violet	7	'V'
Gray	8	'A'
White	9	'W'

Table 1: Resistor Color Codes

The resistance (Ω) value of a resistor can be found using the integer value for each color band as follows:

$$R = (\text{band}_1 \times 10 + \text{band}_2) \times 10^{\text{band}_3}$$

Write a user-modular user friendly C++ program that finds the resistor value from its color code and output the answer as a numerical format. Remember to use function(s)!

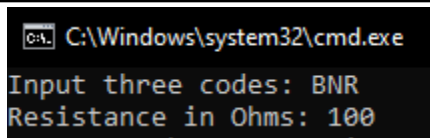
```
// ECE 1310-04
// Author: Jordan Xu
// Description: determining resistance of a band
// Date: 4/9/2021
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int assignCode(char);
int resistance(int, int, int);

int main()
{
    int code1, code2, code3, count;
    code1 = code2 = code3 = 0;
    char currentband;
    cout << "Input three codes: ";
    for (count = 1; count <= 3; count++)
    {
        cin >> currentband;
        switch (count)
        {
            case 1: code1 = assignCode(currentband);
```

```
        case 2: code2 = assignCode(currentband);
        case 3: code3 = assignCode(currentband);
    }
}
if (code1 < 0 || code2 < 0 || code3 < 0)
    cout << "Invalid" << endl;
else cout << "Resistance in Ohms: " << resistance(code1, code2, code3) << endl;
return 0;
}

int assignCode(char x)
{
    int code;
    x = toupper(x); //capitalizes it in code
    switch (x)
    {
        case 'B': code = 0; break;
        case 'N': code = 1; break;
        case 'R': code = 2; break;
        case 'O': code = 3; break;
        case 'Y': code = 4; break;
        case 'G': code = 5; break;
        case 'E': code = 6; break;
        case 'V': code = 7; break;
        case 'A': code = 8; break;
        case 'W': code = 9; break;
        default: code = -1; //makes it invalid
    }
    return code;
}

int resistance(int band1, int band2, int band3)
{
    int variable1 = static_cast<int>(pow(10, band3));
    return (band1 * 10 + band2) * variable1;
}
```



```
C:\Windows\system32\cmd.exe
Input three codes: BNR
Resistance in Ohms: 100
```

2. Create a function that will print the following table to display the sin value and cos value of degrees from 0 to 360 with increments of 10 degrees. Round the value to keep four digits after the decimal point.

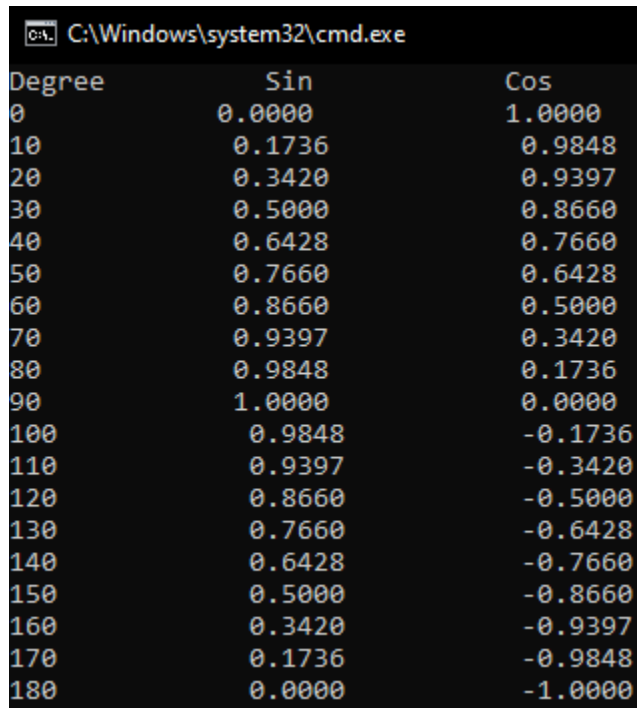
```
// ECE 1310-04
// Author: Jordan Xu
// Description: finding value of sin and cos with degree 0 to 360
// Date: 4/9/2021
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
const double pi = 3.141592653589793238;
void chart();
double Sin(int), Cos(int);

int main()
{
    chart();
    return 0;
}

void chart()
{
    cout << fixed << setprecision(4);
    cout << "Degree" << setw(14) << "Sin " << setw(14) << "Cos" << endl;
    //setw for spacing
    for (int deg = 0; deg <= 360; deg += 10)
    {
        cout << deg << setw(18) << Sin(deg) << setw(18) << Cos(deg) << endl;
        //setw(18) gives formatting issue but fixable with if else statements
    }
}

double Sin(int x)
{
    return sin(x * pi / 180);
}

double Cos(int x)
{
    return cos(x * pi / 180);
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\system32\cmd.exe". The window contains a table of sine and cosine values for degrees from 0 to 180. The table has three columns: "Degree", "Sin", and "Cos". The values are listed in a single column, with the degree value on the left, the sine value in the middle, and the cosine value on the right. The values are rounded to four decimal places.

Degree	Sin	Cos
0	0.0000	1.0000
10	0.1736	0.9848
20	0.3420	0.9397
30	0.5000	0.8660
40	0.6428	0.7660
50	0.7660	0.6428
60	0.8660	0.5000
70	0.9397	0.3420
80	0.9848	0.1736
90	1.0000	0.0000
100	0.9848	-0.1736
110	0.9397	-0.3420
120	0.8660	-0.5000
130	0.7660	-0.6428
140	0.6428	-0.7660
150	0.5000	-0.8660
160	0.3420	-0.9397
170	0.1736	-0.9848
180	0.0000	-1.0000