Lab 8 - Expressions and Math

Directions

• You may not work on assignments during the lab, even if you are done the lab.

1. Expressions

1. Create a new project named lab8. Add a new file named expressions.cpp.

2. As described in the notes, explicit casting changes the type of a value. For example, to cast the variable age to a double, use static_cast<double>(age):
   \[\text{double } x = \text{static\_cast}\text{<double>}(1) / 2; \quad \text{// Evaluates to 0.5}\]

3. Knowing the order of operations and implicit/explicit type casting rules is an important skill to be able to do without relying on a compiler.

   ▪ For each of the expressions below:
     1. Determine (by hand) what value will be stored in the variable.
     2. Explain all implicit casts, explicit casts, overflows, or underflows.

   a) \[\text{double } a = 3.0 + 10 / 20;\]
   b) \[\text{double } b = 3.0 + 10.0 / 20;\]
   c) \[\text{double } c = 3.0 + \text{static\_cast\text{<double>}(10 / 20)};\]
   d) \[\text{double } d = 3.0 + \text{static\_cast\text{<double>}(10)} / 20;\]
   e) \[\text{double } e = 3.0 + 10 / \text{static\_cast\text{<double>}(20)};\]
   f) \[\text{int } f = 3.0 + 10 / \text{static\_cast\text{<double>}(20)};\]
   g) \[\text{int } g = 100000000 * 100000000;\]
      \[\text{// Don't need exact value; just idea of what to expect.}\]
   h) \[\text{double } h = 3.0 + \text{static\_cast\text{<int>}(10.0)} / 20;\]

4. For each of the expressions in the above figure, check your answers using the compiler. Copy and paste this code into a program, and after each statement, output the variable to the screen.

   ▪ Note that the following two statements will generate the output “42”:
     \[\text{cout << 42 \text{<} \text{endl;}}\]
     \[\text{cout << 42.0 \text{<} \text{endl;}}\]

   ▪ To see which numbers are actually floating point values, place the following at the start of your program:
     \[\text{cout << fixed \text{<} \text{setprecision(2)} \text{<} \text{endl;}}\]

   ▪ You will need to #include <iomanip>

   ▪ Note that even if you use fixed and setprecision() that integer values are still printed without any decimal point.
5. Understanding

- What is the difference between floating-point division and integer division. Give an example of each.

- Given the equation
  
  ```csharp
double y = 2 / 5;
```

  state up to four ways you could change the statement, using only implicit or explicit casting, to make \( y \) hold the value 0.4.

2. Pythagorean Theorem

Recall that for a right triangle with sides of length \( a \) and \( b \), and hypotenuse of length \( c \); \( c \) can be found by using the equation \( a^2 + b^2 = c^2 \).

Write a program that asks the user to enter the lengths of the two sides (\( a \) and \( b \)) of the triangle, and displays the length of the hypotenuse.

Here are some useful functions; you’ll need to `#include <cmath>`

- `pow()` to calculate the squares.
- `sqrt()` to calculate the square root.
- `asin()` to calculate arc-sin.

Have the program print out (rounded to one decimal) the values of all three angles:

- In a right triangle the `sin()` of an angle equals the length of the opposite side divided by the length of the hypotenuse.

- In the figure, we have [math, not C++]: \( \sin(k) = b / c \)
  
  So angle \( k = \text{arc-sin} \ of \ b / c \)
  
  In C++, we write:
  
  ```c
  double angleK_radians = asin(b/c);
  ```

  The result will be in radians. Multiply by 57.295 to convert to degrees.

  *Hint: Isn't this number magical!*

Finally, print the sum of all three angles.

**Example output:**

Enter the lengths of the base and height of a right-triangle.

\(: 5.7 \ 12.4 \)

Hypotenuse is: 13.6
Angle 1: 24.7
Angle 2: 65.3
Angle 3: 90.0 (of course!)
Sum of angles: 180.0
3. ASCII Table

Add to your expressions.cpp program to generate the following ASCII table:

```
ASCII Table:
************
= 32   != 33   "= 34   #= 35   %= 36   ^= 37   &= 38   '= 39
(= 40   )= 41   *= 42   += 43   -= 44   *= 45   /= 47
0= 48   1= 49   2= 50   3= 51   4= 52   5= 53   6= 54   7= 55
8= 56   9= 57   := 58   ;= 59   <= 60   == 61   >= 62   ?= 63
@= 64   A= 65   B= 66   C= 67   D= 68   E= 69   F= 70   G= 71
H= 72   I= 73   J= 74   K= 75   L= 76   M= 77   N= 78   O= 79
P= 80   Q= 81   R= 82   S= 83   T= 84   U= 85   V= 86   W= 87
X= 88   Y= 89   Z= 90   [= 91   ]= 92   ^= 94   _= 95
`= 96   a= 97   b= 98   c= 99   d=100   e=101   f=102   g=103
h=104   i=105   j=106   k=107   l=108   m=109   n=110   o=111
p=112   q=113   r=114   s=115   t=116   u=117   v=118   w=119
x=120   y=121   z=122   %=123   |=124   }=125   ~=126
```

Hints

- Use a for loop! For each value, print out the value as a character and an integer.
  - With a variable char ch = 'a'; you can display the character as a symbol with:
    
    ```cpp
    cout << ch;
    ```
  - With a variable char ch = 'a'; you can display the number by casting it to an int.
- Use column-based formatting (like `setw(3)` ) for spacing in the table.
- Add line-feeds after every N'th element to generate new lines. What operator works best?

4. Extra Challenge: Temperature Conversion

Try this task for an extra challenge.

- Write a program which converts temperatures from Celsius to Fahrenheit.
  - The conversion formula (where F is degrees Fahrenheit, and C is degrees Celsius) is:
    
    $F = \frac{9}{5} C + 32$

- Your program should prompt the user for the current temperature and then convert it.
- Display all temperatures to one decimal place.
- Sample interaction (may look different if you choose):
  
  ```cpp
  Please enter the temperature in degrees C: 20.1
  20.1'C equates to 68.2'F.
  ```

- Ensure your code has comments and uses named constants (no magic numbers!).
- Modify your temperature converter to also convert the temperature into degrees Kelvin [K]:
  
  $K = C + 273.15$

- Modify your program to also convert the temperature into degrees Rankine [R]:
  
  $R = \frac{9}{5} K$

Understanding

- How do order of operations and implicit/explicit type casting work in your program?