1 Template:

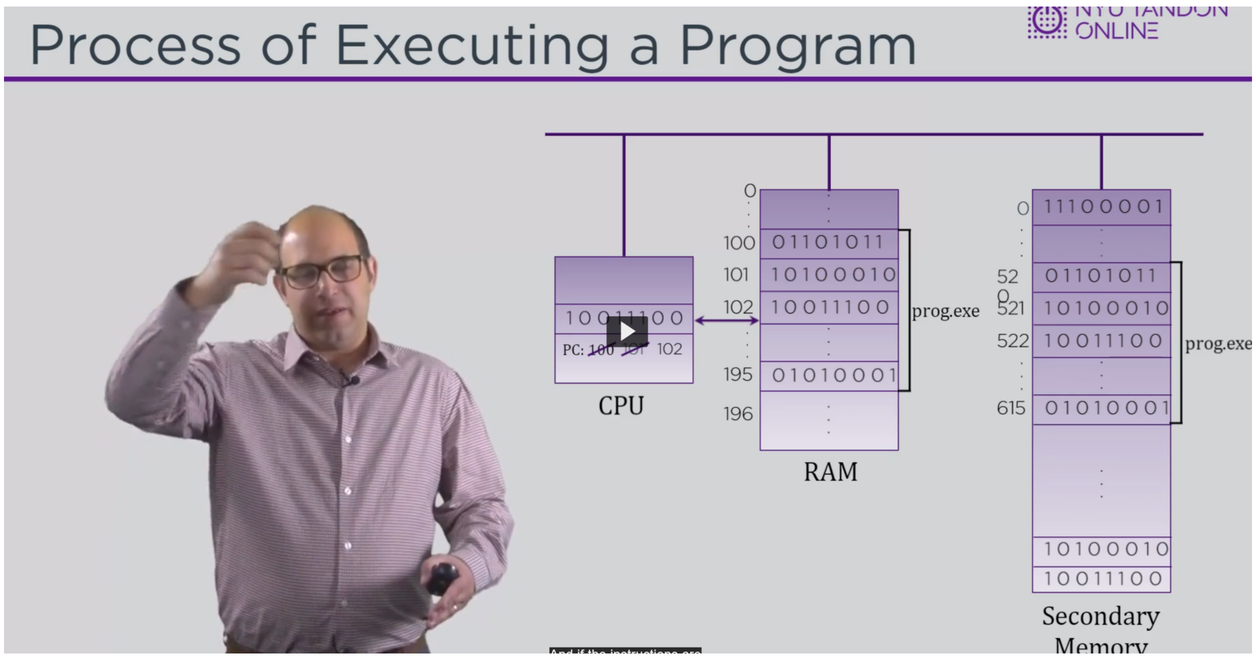
```
#include <bits/stdc++.h>

using namespace std;

int main()

{

// solution comes here

}
```

This library bits/stdc++/h can be used to replace iostream, vector, and string

How to run c++ code: If you name your file test.cpp

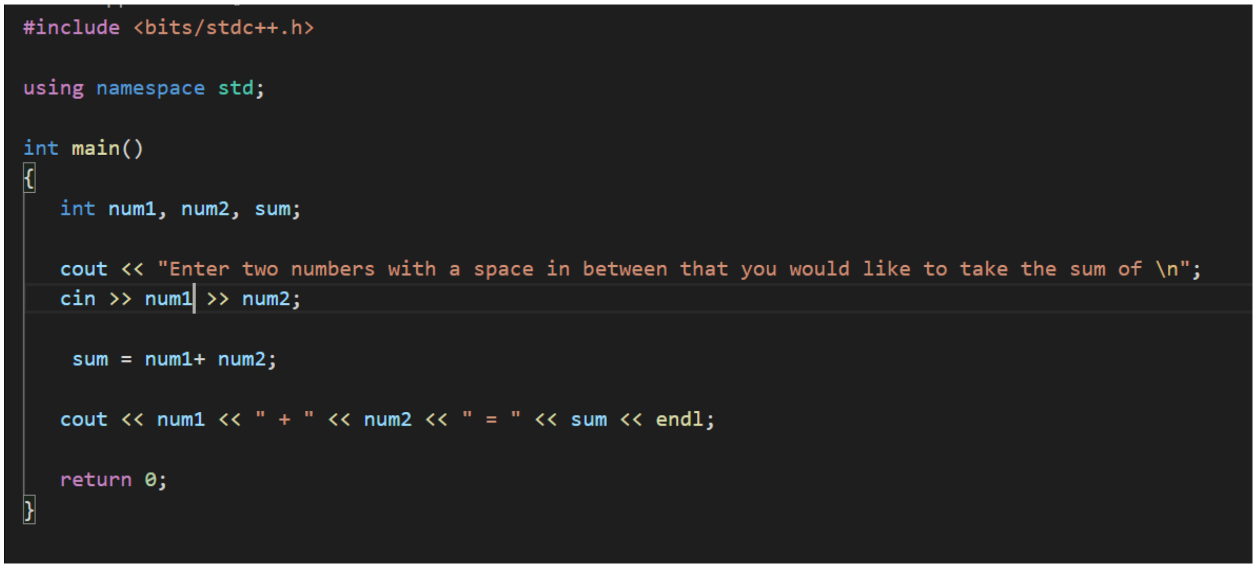
```
-std=c++11 -O2 -Wall test.cpp -o test        or        g++ -o test test.cpp

                                                        ./test
./ test
```

<u>2 Concept of CPU, RAM, and Secondary memory</u>



The secondary memory is what stores the data types. Each column represents a byte and each byte contains 8 bits. Certain number of columns can represent the executable file .exe. That is, the file after the programme has been compiled. That executable file goes to the temporary random access memory and is controlled by the CPU in interpreting those binary numbers.

Problem 0: Basic sum program



```
#include <bits/stdc++.h>

using namespace std;

int main()
{
   int num1, num2, sum;

   cout << "Enter two numbers with a space in between that you would like to take the sum of \n";
   cin >> num1 >> num2;

    sum = num1+ num2;

   cout << num1 << " + " << num2 << " = " << sum << endl;

   return 0;
}
```

This program outputs the sum of two integers. This program also emphasizes data types
For int, it takes up 32 bits worth of information on the secondary memory, that is 4 bytes, and in every byte there are 8 bits, so 8 times 4 is 32 and every byte is represented as a column in the secondary application which can make up an entire application.

3 Max value of int data type
To illustrate the max value of the int data type, I inputted 2^32 + 2, and the compiler did not like that so it just did the calculation as 2^32 +0. This emphasizes that the max value of int is 2^n, where n is the total number of bits the data type can occupy.

Problem1: days conversion
Write a program so that the user can input the number of days they were traveling during the holiday break, and print out the information in weeks and additional days.

Solution:
Created variables for the number of days the user would input, weeks for calculating the number of weeks given how many days the user would input, and days left was the calculation for the remaining days that were left after the initial weeks calculation.
This is a simple division and remainder problem, and you need to use the following concepts:

1. Integer data type
2. Inputting and storing data with cout and cin commands
3. Operators like division and mod(remainder operator)

After creating the variables I outputed a string that asked the user the number of days that they traveled and stored it in the days variable with the cin command

Then I did the calculations with dimensional analysis(conversion)

7 days = 1 week

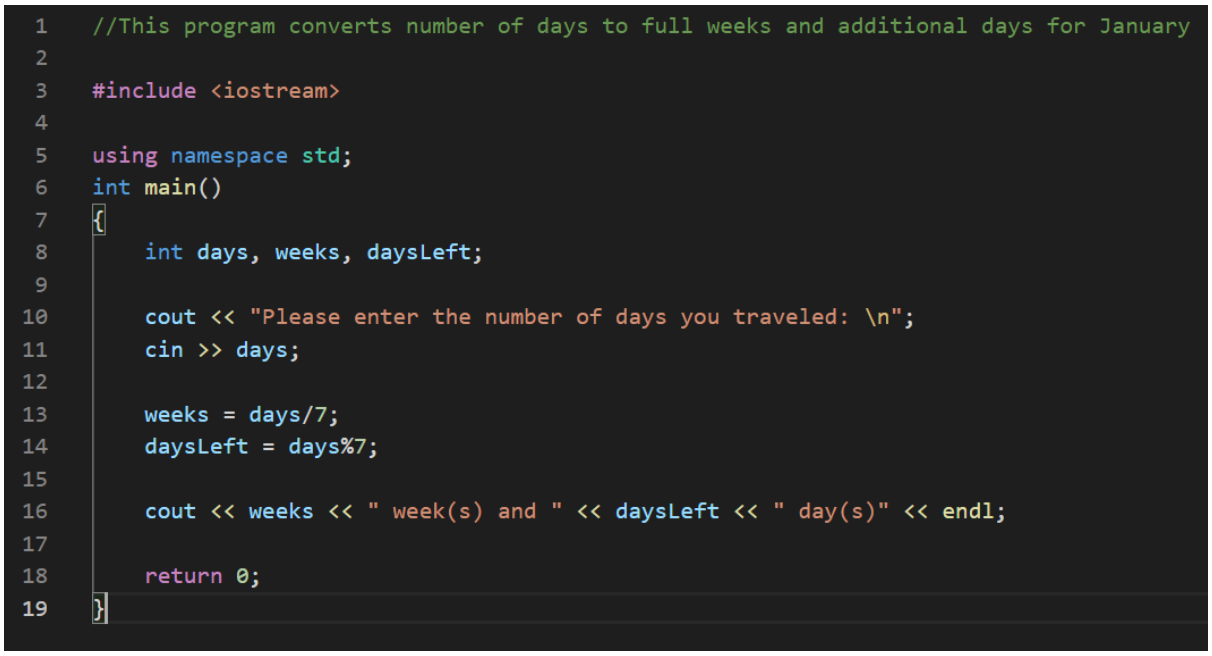
So the calculation would be n days * 1weeks/7 days.

Doing it this way made me go back to the fundamental engineering concepts I learned in undergrad, units conversion. I was making the mistake of dividing days by 4, confused that dividing by four would converted months to weeks, not days to weeks or vice versa for the former.

daysLeft was the variable I made to store the number of days left after dividing. Like in elementary school, we learned to divide numbers and the number that was left at the bottom after finishing the calculations, was the remainder.

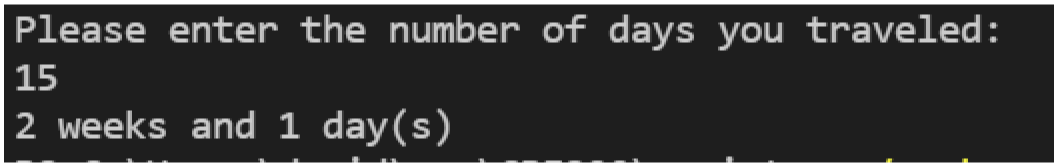
That is what the mod operator does, it takes the remainder of the division, so if we wanted to divide the days by 7, we should also do days ‘mod’ 7, or days%7;

Finally I printed the the weeks and the daysLeft variables in a way that a normal user can understand



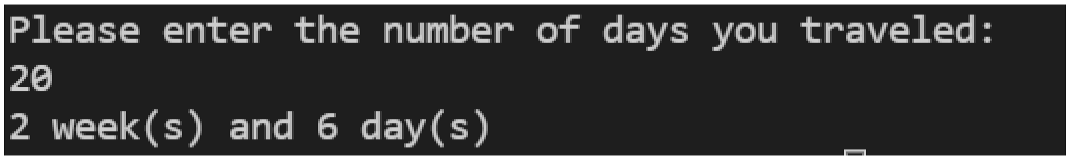
```cpp
//This program converts number of days to full weeks and additional days for January

#include <iostream>

using namespace std;
int main()
{
    int days, weeks, daysLeft;

    cout << "Please enter the number of days you traveled: \n";
    cin >> days;

    weeks = days/7;
    daysLeft = days%7;

    cout << weeks << " week(s) and " << daysLeft << " day(s)" << endl;

    return 0;
}
```

Results:
Trial 1



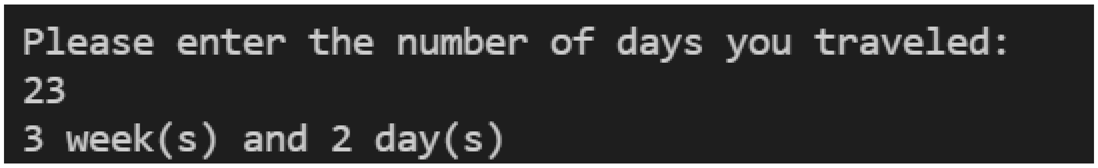
```
Please enter the number of days you traveled:
15
2 weeks and 1 day(s)
```

Trial 2



```
Please enter the number of days you traveled:
20
2 week(s) and 6 day(s)
```

Trial 3



```
Please enter the number of days you traveled:
23
3 week(s) and 2 day(s)
```

Modified the code each time to make the gui prettier and make more sense to the user with no programming knowledge.

4 Float and Double data types.
Float can hold up to 4 columns of memory while double can hold double of that, which is 8.
So float can hold 32 bits or 4 bytes, while double can hold 64 bits or 8 bytes. Both of them are used for decimal or integer values, except if an integer is used, it will print out for example, 2.0, not 2.

Problem 2: Area of a circle
Calculate the area of a circle after the user enters a radius of any size

Solution:
First I outputted a please enter radius prompt, and stored it in the rad variable using the double data type, in case the user enters a decimal that is longer than the allowed bits stored in a float. I also created an Area variable to store the calculation of the area of a circle.

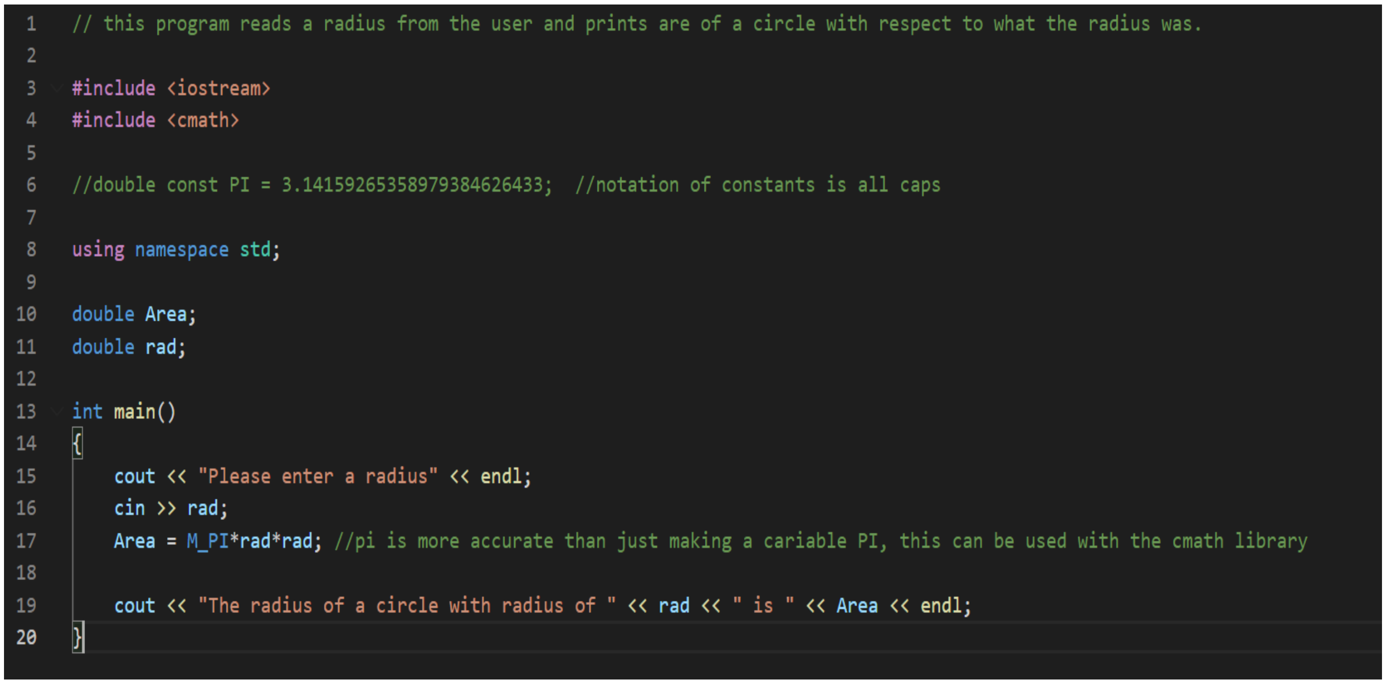
Then, I used the Area of a circle formula:

$$A = \pi * r^2$$

In code this represents:
Area = M_PI*rad*rad;

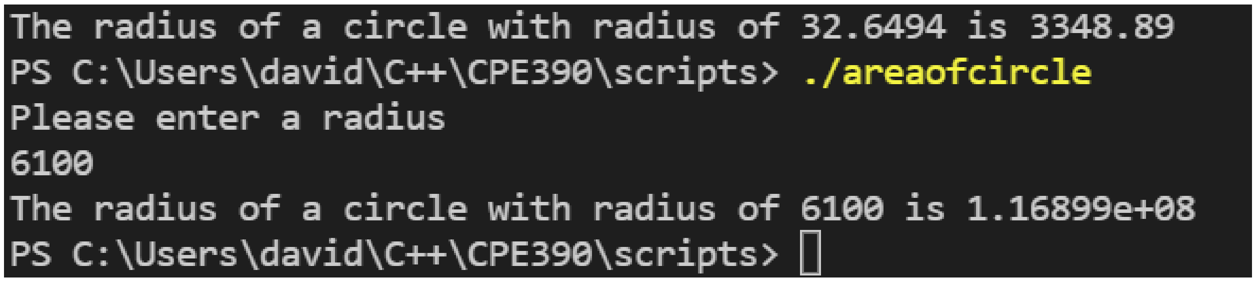
M_PI is a constant variable stored in the cmath formula, this library allows you to enter the pi variable without any errors.

Another way to add the pin variable is to just make a constant variable pi and approximate it to the nearest pi digit that you know, but this will still not be as accurate as using the cmath library pi variable.



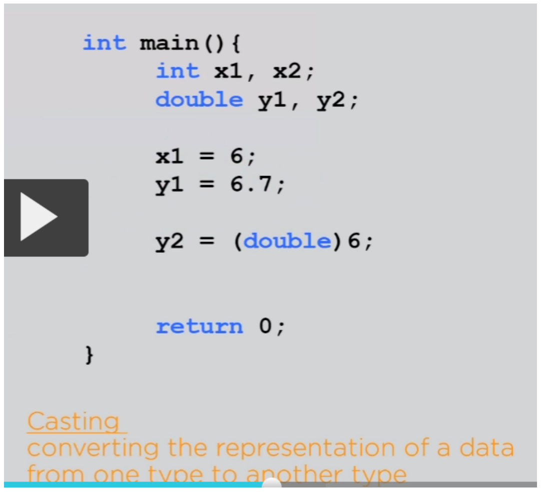
```cpp
// this program reads a radius from the user and prints are of a circle with respect to what the radius was.

#include <iostream>
#include <cmath>

//double const PI = 3.14159265358979384626433;  //notation of constants is all caps

using namespace std;

double Area;
double rad;

int main()
{
    cout << "Please enter a radius" << endl;
    cin >> rad;
    Area = M_PI*rad*rad; //pi is more accurate than just making a cariable PI, this can be used with the cmath library

    cout << "The radius of a circle with radius of " << rad << " is " << Area << endl;
}
```

Results



```
The radius of a circle with radius of 32.6494 is 3348.89
PS C:\Users\david\C++\CPE390\scripts> ./areaofcircle
Please enter a radius
6100
The radius of a circle with radius of 6100 is 1.16899e+08
PS C:\Users\david\C++\CPE390\scripts> 
```

5 Typecasting



```
int main(){
      int x1, x2;
      double y1, y2;

      x1 = 6;
      y1 = 6.7;

      y2 = (double)6;


      return 0;
}
```

Casting
converting the representation of a data from one type to another type

Typecasting can be used when you improperly assign a wrong data type to a number or character. To fix this, one must write the correct data type in a parenthesis before using the number or character. In this example, they are converting the originally integer 6 into double format, which happens to be a decimal.

Implicit casting: C++ will always calculate a value to its most accurate form

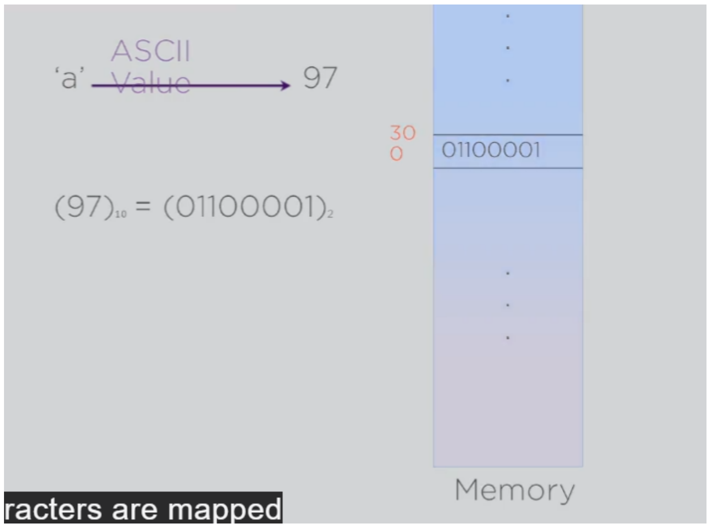
For example, if you divide a decimal by a integer, the result will be a decimal in order to not lose any accuray.

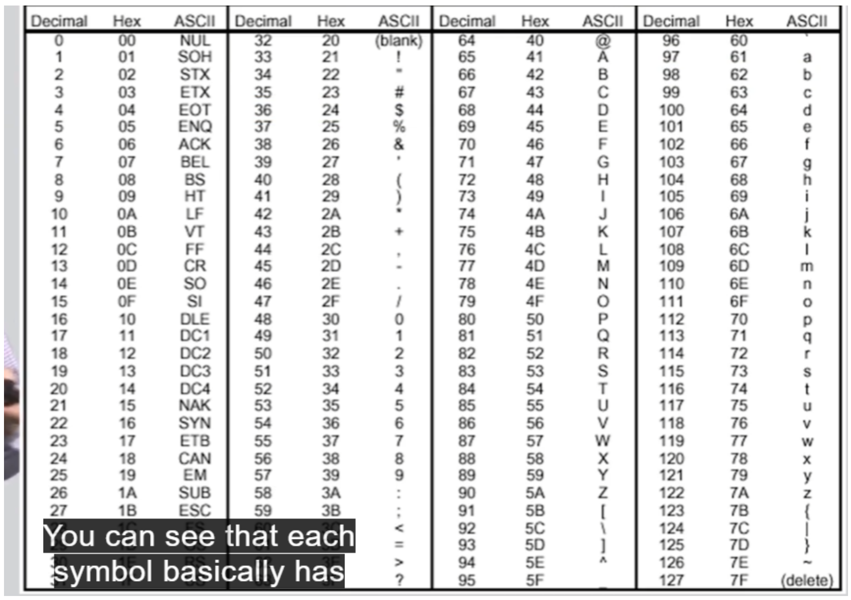
The char data type: Allows you to store letters in code.

The inner representation of what happens in the computer:

Each char data uses 1 byte(8 bits). 2^8 = 256

Ascii maps each letter to a value in the memory in the form of hexadecimal, binary, or decimal form.





Decimal	Hex	ASCII	Decimal	Hex	ASCII	Decimal	Hex	ASCII	Decimal	Hex	ASCII
0	00	NUL	32	20	(blank)	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	<	92	5C	\	124	7C	|
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	=	93	5D	]	125	7D	}
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	>	94	5E	^	126	7E	~
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	?	95	5F	_	127	7F	(delete)

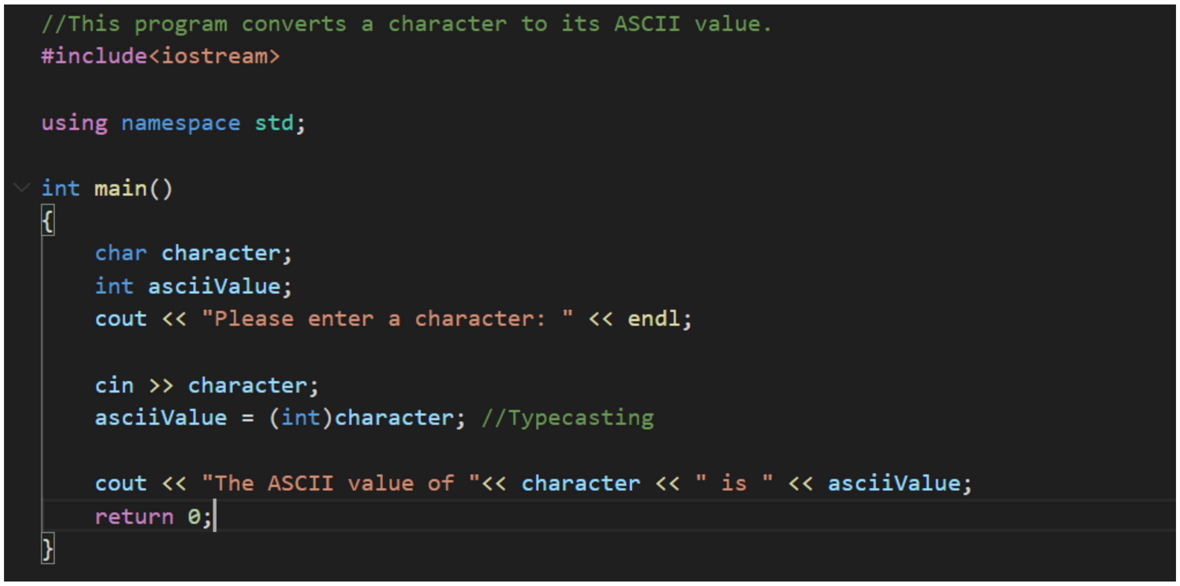
You can see that each
symbol basically has

Problem 3: What is my ASCII value?

Write a program that reads from the user a single character, and prints it's ASCII value.

Example:
Please enter a character:
T
The ASCII value of T is 84

Solution:
Similar to other problems, with only one difference. The formula used was asciiValue = (int)character. I used the character variable for the input character and asciiValue to store the input character as its ascii value. Since all ascii values in the computer are hardcoded into the memory, it was easy just to equate the two values and turn the character into an int via typecasting, otherwise there will be an error.



```cpp
//This program converts a character to its ASCII value.
#include<iostream>

using namespace std;

int main()
{
    char character;
    int asciiValue;
    cout << "Please enter a character: " << endl;

    cin >> character;
    asciiValue = (int)character; //Typecasting

    cout << "The ASCII value of "<< character << " is " << asciiValue;
    return 0;
}
```

Results:

```
Please enter a character:
a
The ASCII value of a is 97
```

Had to use a different compiler because my compiler was giving me issues even though the code was correct.

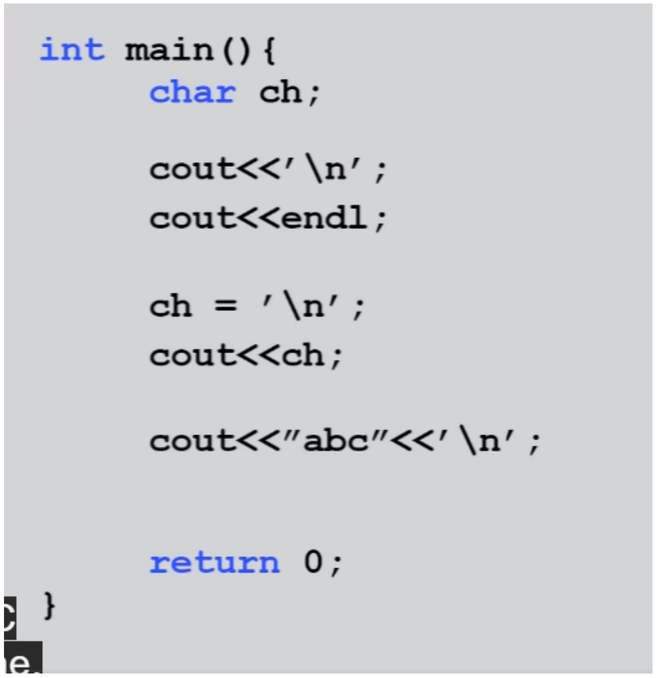
6 Assigning characters to the char data type

Must use ‘ ’ around the letter that you want to be assigned

For example:

If you want to assign the letter a to a variable of data type char:
Char var = ‘a’;
And this is stored into the variable var for later use.
Other literals can include ‘B’ ‘3’ ‘$’

New line character: brakes the line ‘\n’ or “/n”



```
int main(){
     char ch;

     cout<<'\n';
     cout<<endl;

     ch = '\n';
     cout<<ch;

     cout<<"abc"<<'\n';


     return 0;
}
```

Literals manipulation

If you want to get the ascii of the one next to the letter that you input then you must add that literal by 1

```
int main()
{

ch1  = 'a'
ch2 = 'a' + 1;


cout << ch2<< end;
cout << 'a' + 1<< endl;
cout << (char)('a' + 1)<< endl;          // changes the number that represents a+1 and changes it to char
return 0;
}
```

Problem 4: Convert to upper case

Write a program that reads from the user a lower case letter, and prints it's corresponding upper case letter.

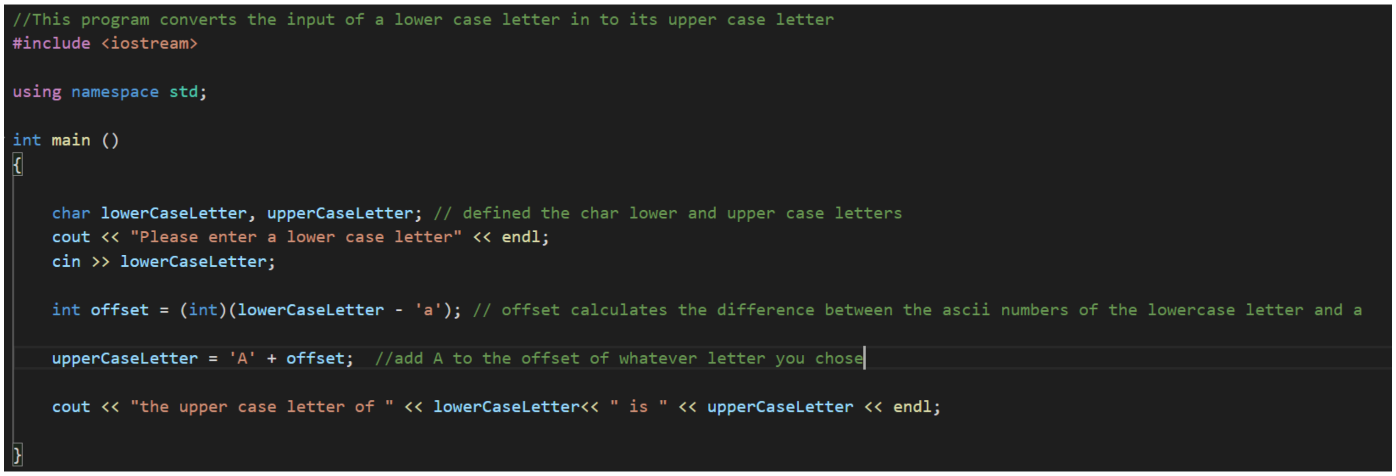
Example:
Please enter a lower case letter:
t
The upper case of t is T

Solution:

This problem was a little tricky. You have to know that ‘a’ means the ascii value and when you write (int)’a’, that is the number of ascii values of a.

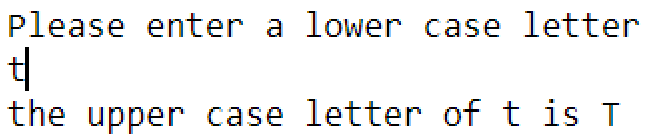
First you need to convert and make the offset. The offset is the difference between the start of the table, ‘a’, and the lowercase number you chose, lets say t.

Now you have to add that offset to the upper case value ‘A’ to get the anwer



```cpp
//This program converts the input of a lower case letter in to its upper case letter
#include <iostream>

using namespace std;

int main ()
{

    char lowerCaseLetter, upperCaseLetter; // defined the char lower and upper case letters
    cout << "Please enter a lower case letter" << endl;
    cin >> lowerCaseLetter;

    int offset = (int)(lowerCaseLetter - 'a'); // offset calculates the difference between the ascii numbers of the lowercase letter and a

    upperCaseLetter = 'A' + offset;  //add A to the offset of whatever letter you chose

    cout << "the upper case letter of " << lowerCaseLetter<< " is " << upperCaseLetter << endl;

}
```

Results:

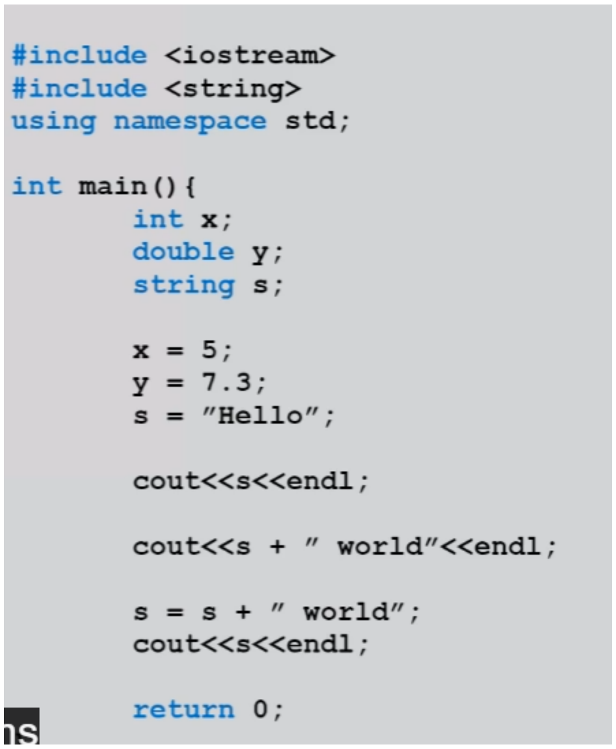


```
Please enter a lower case letter
t
the upper case letter of t is T
```

7 Strings and literals:

A string is a text, or sentence while literals are a sequence of characters
To use the strings data type, add a new library called <string> with #include <string>

Example



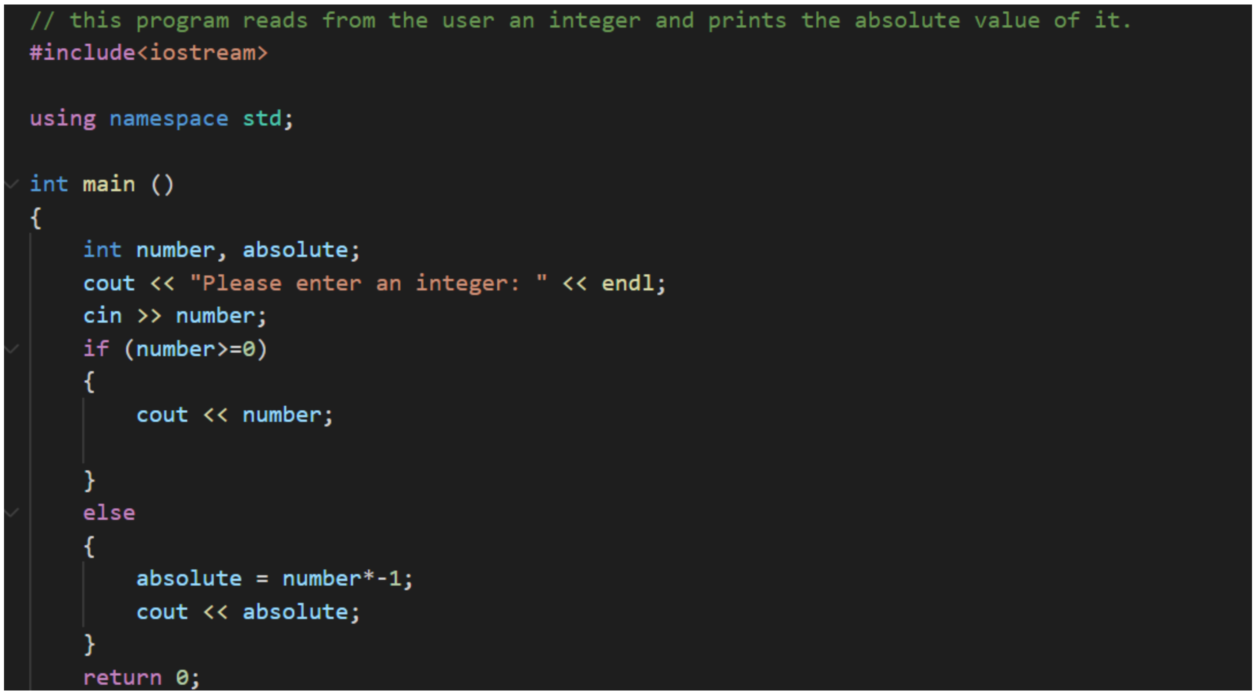
```
#include <iostream>
#include <string>
using namespace std;

int main(){
        int x;
        double y;
        string s;

        x = 5;
        y = 7.3;
        s = "Hello";

        cout<<s<<endl;

        cout<<s + " world"<<endl;

        s = s + " world";
        cout<<s<<endl;

        return 0;
```

8 Branching statements

```
If(condition)
{
        code happens if the condition is met
}

else
{
        This code will happen if the condition above is not met
}
```

Problem 5: absolute value

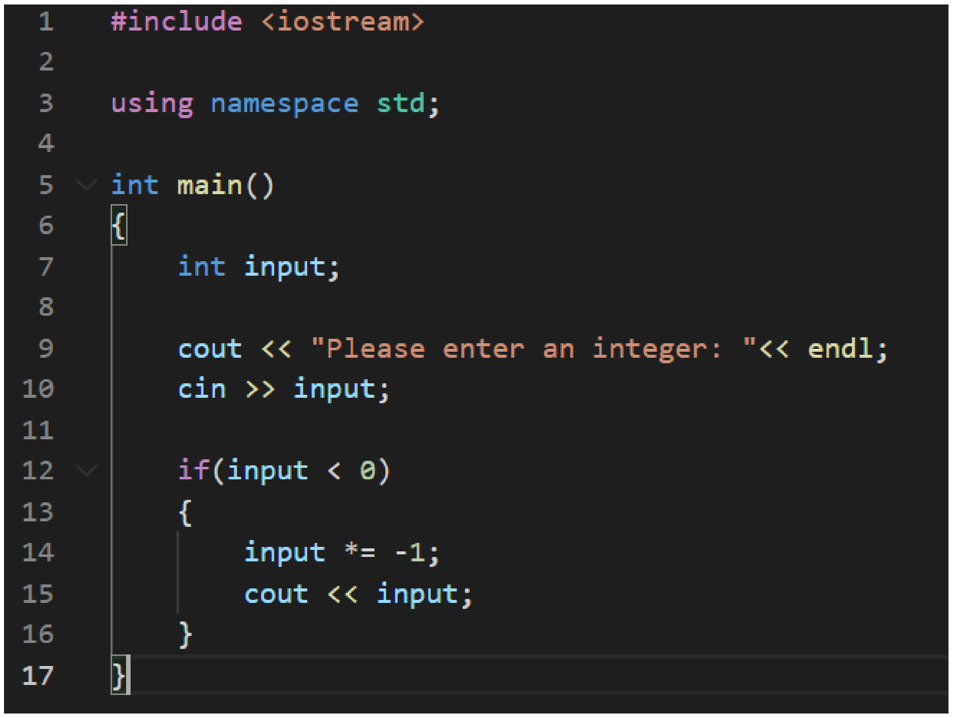
Initial thought about this was to make the user input a number and if the number was positive, the number would print out, else the number times -1 would print, storing that into a new variable.
(20 lines)



```cpp
// this program reads from the user an integer and prints the absolute value of it.
#include<iostream>

using namespace std;

int main ()
{
    int number, absolute;
    cout << "Please enter an integer: " << endl;
    cin >> number;
    if (number>=0)
    {
        cout << number;

    }
    else
    {
        absolute = number*-1;
        cout << absolute;
    }
    return 0;
```

A more efficient way was found, where there are less lines of code and less variables used.

This more efficient way of solving the problem, took the input of the user, and if that input was less than 0, it would point out that number times -1 and store it in that same variable. (17 lines)



```cpp
#include <iostream>

using namespace std;

int main()
{
    int input;

    cout << "Please enter an integer: "<< endl;
    cin >> input;

    if(input < 0)
    {
        input *= -1;
        cout << input;
    }
}
```

input *= -1 is just input = input * -1

<u>9 Set precision</u>

Using library <iomanip>
```
cout<< setprecision(decimal place) << variable;
```

This is used to write your answers to the amount of decimal places you want. For example, if you want to round a decimal to 2 decimal places inside a variable sum then you must do:

```
cout<< setprecision (2) << sum;
```