# Additional Tasks – NOT MANDATORY

# C++

The C++ code in the last tutorial converted from and to unsigned binary. Have a look at the code below for signed binary. Can you see the difference? Try it out and watch the accompanying videos discussing how they work.

#include <iostream>

#include <string>

using namespace std;

int sBinToDec() {

cout << "Hello, please enter a binary number, with no spaces.";

//read in the string so we can parse through it, character by character.

string binaryNumber;

getline(cin, binaryNumber);

//get the length of the string, then create a variable called 'currentBit' that will point

//to the last bit

int stringLength = binaryNumber.length();

int currentBit = stringLength - 1;

//the variable bitValue will start at 1 and double each time we move to the next bit along

int bitValue = 1;

//decimalOutput will keep a total and will add up to the result.

int decimalOutput = 0;

//once currentBit has gone past the first character in the string, we stop.

while (currentBit > -1) {

/\*

\* Here, we read the current bit, and multiply it by the value of the column we are in.

\* Because we are reading a character, C++ converts it to it's ASCII equivalent, so we

\* subtract '0' to get the integer value 0 or 1.

\*/

//if we are on the last column, then the value should be a negative number

if (currentBit == 0) {

bitValue = bitValue \* -1;

}

decimalOutput += (bitValue \* (int)(binaryNumber[currentBit] - '0'));

//move to the next bit along, where the value is double the last column.

bitValue = bitValue \* 2;

--currentBit;

}

return decimalOutput;

}

string decToSBin() {

cout << "Hello, please enter a decimal number, with no spaces.";

//read in the string and convert it to an integer.

int decimalNumber;

cin >> decimalNumber;

string binaryOutput = "";

bool isNegative = false;

if (decimalNumber < 0) {

isNegative = true;

//we need to convert it as a positive number

//then afterwards flip the bits and add one.

//if we subtract one from the positive value,

//this has the same effect

decimalNumber = (decimalNumber \* -1) - 1;

}

//while the number is greater than 1, we divide it by 2.

while (decimalNumber > 0) {

//dividing an integer by two gives us a whole number answer, not a fraction.

//so we work out the remainder first using % operand

int remainder = decimalNumber % 2;

//then we divide the number by two for the next step.

decimalNumber = decimalNumber / 2;

//now we add the remainder to our binary output.

binaryOutput = binaryOutput + to\_string(remainder);

}

//add a zero to the end because we're dealing with signed binary now

binaryOutput = binaryOutput + "0";

//let's group this into 8 bits - this loop will add zeros until we have a

//multiple of 8 digits in our output.

while (binaryOutput.length() % 8 != 0) {

binaryOutput = binaryOutput + "0";

}

//now we have to read the remainders backwards for the output.

reverse(binaryOutput.begin(), binaryOutput.end());

//flip the bits if the number was originally negative

if (isNegative) {

int index = 0;

//loop through and turn all 0s to 1s and vice versa

while (index < binaryOutput.length()) {

if (binaryOutput[index] == '0') {

binaryOutput[index]++;

} else {

binaryOutput[index]--;

}

index++;

}

}

return binaryOutput;

}

int main() {

//comment out the conversion you don’t want to perform

cout << sBinToDec();

cout << decToSBin();

}

PYTHON  
  
The Python code in the last tutorial converted from and to unsigned binary. Have a look at the code below for signed binary. Can you see the difference? Try it out and watch the accompanying videos discussing how they work.

def decToSBin(decString):  
 # we can use the repeated division to calculate this.  
 # first convert the string to a decimal number  
 decimalNumber = int(decString)  
 outputString = ""  
  
 isNegative = False  
 if decimalNumber < 0:  
 isNegative = True  
 # we need to convert it as a positive number  
 # then afterwards flip the bits and add one.  
 # if we subtract one from the positive value,  
 # this has the same effect  
 decimalNumber = (decimalNumber \* -1) - 1;  
  
 # we keep dividing until we end up with 1/2  
 while(decimalNumber > 0):  
 # % (modulus) returns the remainder when dividing so add that to the binary string  
 outputString += str(decimalNumber%2)  
  
 # now we have the remainder, we just want to do the division,  
 # but subtract 1 if the number is odd so we end up with a whole number  
 decimalNumber = int(decimalNumber / 2)  
  
  
 # add a zero to the end because we're dealing with signed binary  
 outputString += "0"  
  
  
  
 # this needs to be grouped into 8 bits  
 while len(outputString) % 8 != 0:  
 outputString += "0"  
  
  
 # finally, if we were originally dealing with a negative number, flip the bits.  
 if(isNegative):  
 output = ""  
 index = 0  
 while index < len(outputString):  
 if outputString[index] == "0":  
 output += "1"  
 else:  
 output += "0"  
 index = index + 1  
 else:  
 output = outputString  
  
 # lets reverse the string to return it  
 return output[::-1]  
  
  
def sBinToDec(binString):  
 # we could validate the string here to make sure it's only 1s and 0s  
  
 # we want to work from the right hand side of the string to the left  
 # the first value will be worth 1 so have a variable for that  
 columnValue = 1  
 numberValue = 0  
  
 # now loop from the length of the string to zero, subtracting by 1 each time  
 for i in range(len(binString),0,-1):  
 # check if this is the last column, if so, make it negative  
 if i == 1:  
 columnValue = columnValue\*-1  
  
 # if there is a 1, add the column value to the total  
 if(binString[i-1] == "1"):  
 numberValue += columnValue  
  
 # increase the column value for the next one along  
 columnValue \*= 2  
  
 # return the value now we've calculated it  
 return numberValue  
  
# main part of the program here. Ask the user to make a choice on conversion  
userChoice = input("Enter 1 to conver from binary to decimal, or 2 to convert decimal to binary")  
  
# either way we need a number input  
numberToConvert = input("Enter the number you want to convert")  
  
# convert it according to the choice the user made  
if(userChoice == "1"):  
 print(sBinToDec(numberToConvert))  
else:  
 print(decToSBin(numberToConvert))