C++ Programming Logbook
Joseph Butterworth

Contents

Hello World - 02/02/21
https://stackoverflow.com/questions/9935027/cout-does-not-name-a-type
https://stackoverflow.com/questions/28236870/undefined-reference-to-stdcout

Coding Hello world is different in C++ to C as it uses a different header.

<iostream> reads inputs and writes outputs to terminal, or to file. It has different namespaces for the commands. When using the same namespace across the entire program it is easier to include the using namespace command. When using different namespaces the command can be written as:

std::cout << “Hello world!\n”;

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

int main()
{
    cout << "Hello world!\n";
}
```

When compiling I initially had issues, a c++ program is compiled using the g++ command instad of gcc.

```
PS G:\1204_Advanced_Programming> gcc .\hello_world.cpp -o hello
C:\Users\ludde\AppData\Local\Temp\ccUEMhQQ.o:hello_world.cpp:(.text+0x19):
undefined reference to `std::cout'
C:\Users\ludde\AppData\Local\Temp\ccUEMhQQ.o:hello_world.cpp:(.text+0x1e):
undefined reference to `std::basic_ostream<char, std::char_traits<char> >&
std::operator<< <std::char_traits<char> >(std::basic_ostream<char,
std::char_traits<char> >&, char const*)'
C:\Users\ludde\AppData\Local\Temp\ccUEMhQQ.o:hello_world.cpp:(.text+0x35):
undefined reference to `std::ios_base::Init::~Init()'
C:\Users\ludde\AppData\Local\Temp\ccUEMhQQ.o:hello_world.cpp:(.text+0x56):
undefined reference to `std::ios_base::Init::Init()'
collect2.exe: error: ld returned 1 exit status
PS G:\1204_Advanced_Programming> g++ .\hello_world.cpp -o hello
PS G:\1204_Advanced_Programming> .\hello.exe
Hello world!
```

P1: Solving Sudoku
https://en.wikipedia.org/wiki/Sudoku_solving_algorithms
https://www.geeksforgeeks.org/sudoku-backtracking-7/
https://www.tutorialspoint.com/cplusplus/cpp_multi_dimensional_arrays.htm

Algorithm Design - 02/02/21

1) What are the various types of sudoku solving algorithms available? (List at least 2)

For solving sudoku there are two main methods:

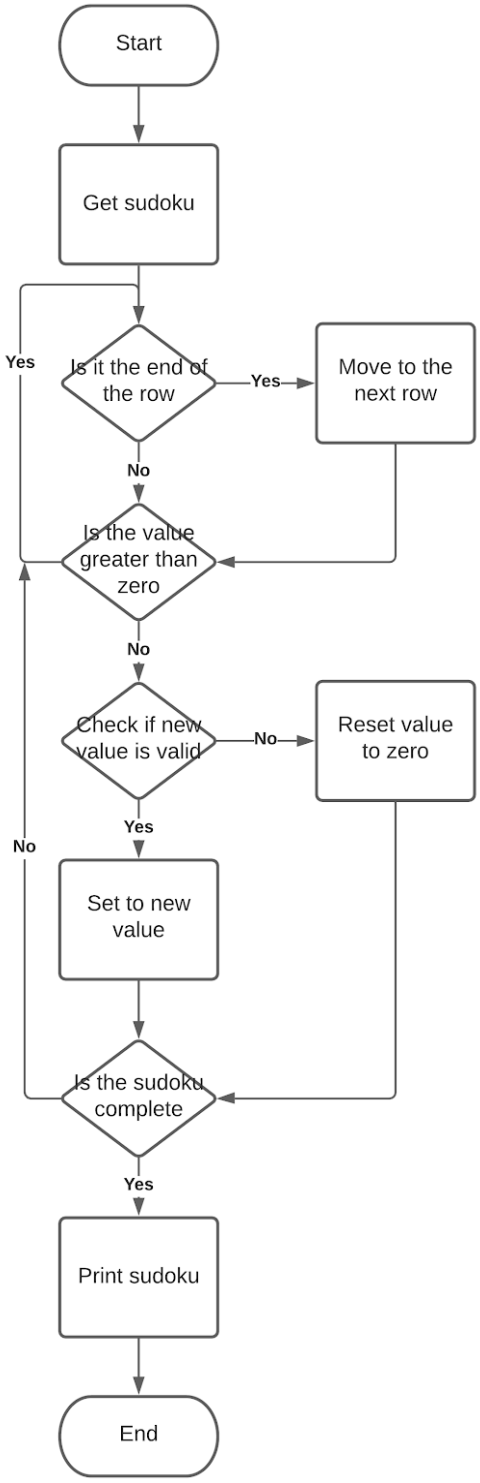
Backtracking is a type of depth-first search algorithm that is looking for the solution using recursion. It will look down the branch until an error is detected and then backtrack till the last split.

Stochastic searches randomly assigns numbers to the cells and then checks for shuffles that reduces the number of total errors. It will keep iterating this until a global maximum is reached, i.e. the sudoku is solved.

2) Work together with your partners to decide on the best approach to solve the problem, then, choose one algorithm that you plan to implement in this lab and list down its pros and cons.

I will be implementing a backtracking algorithm. The advantages of this method are the solution is guaranteed and the algorithm is simpler to implement. The disadvantage of this method is the program will be comparatively slow to other algorithms.

3) Then, draw a suitable program flowchart to illustrate your approach in your logbook. You may draw multiple flowcharts to explain your program's subroutines or functions if needed.



File Handling - 02/02/21

4) How do you read a character from file in C++?

To read from a file in C++ the <fstream> and <string> headers are needed.

The code will read in the file one word at a time and then print it out to the terminal.

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

int main()
{
    fstream inputStream;
    string text;

    inputStream.open("lazy_doggo.txt");

    while(!inputStream.eof())
    {
        inputStream >> text;
        cout << text << " ";
    }

    return 0;
}
```

5) How do you read a character from file in C?

To read in a file using c the fgetc command is used on characters and the fgets command is used on strings.

```c
#include <stdio.h>

int main()
{
    FILE *fp;

    fp = fopen("test.txt", "r");
    while (!feof(fp))
    {
        char letter = fgetc(fp);
        printf("%s", letter);
    }
    fclose(fp);
}
```

6) Will your answer for 5) work in a C++ code?

My code for part 5 should work in C++ as all C features should work. However, it is recommended against using C libraries as there is the potential to conflict with C++ libraries.

File Reading - 03/02/21

To read in the file I used the same method as in my preparation. However, I also added some processing to convert the value read in from a character to an integer value.

```
int *import_board(void)
{
    fstream inputStream;
    char input;
    int number;
    int row = 0;
    int col = 0;
    inputStream.open("SUDOKU.txt");

    while (!inputStream.eof())
    {
        inputStream >> input;

        //convert character into number
        number = input - 48;
        if (number < 0 || number > SIZE)
        {
            number = 0;
        }

        //update row position
        if (col == SIZE)
        {
            row++;
            col = 0;
        }

        //set position to number
        grid[row][col] = number;
        col++;
    }

    return (int *)grid;
}
```

Sudoku Display - 03/02/21

To print out the board that I read in I created a function, display_board(). display_board() also creates a grid around the sub-boards. This is an aesthetic change, but it will make it easier for the user of the program to interpret the sudoku board.

```
void display_board(int *board)
{
    for (int row = 0; row < SIZE; row++)
    {
        if (row % SUBSIZE == 0)
        {
            cout << " ------- ------- ------- " << endl;
        }

        for (int col = 0; col < SIZE; col++)
        {
            if (col % SUBSIZE == 0)
            {
                cout << "| ";
            }

            int boardPos = (SIZE * row + col);
            if (board[boardPos] == 0)
            {
                cout << "_ ";
            }
            else
            {
                cout << board[boardPos] << " ";
            }
        }
        cout << "|" << endl;
    }

    cout << " ------- ------- ------- " << endl;
}
```

```
PS G:\1204_Advanced_Programming\P1_Solving_Sudoku\3.2_Display> g++
.\Sudoku_solver.cpp .\Sudoku_solver.h -o sudoku
PS G:\1204_Advanced_Programming\P1_Solving_Sudoku\3.3_Display> .\sudoku.exe
 ------- ------- -------
| _ 2 _ | 8 _ _ | _ _ _ |
| 9 _ 1 | _ _ _ | _ _ _ |
| _ 7 _ | 2 4 6 | _ _ _ |
 ------- ------- -------
| 2 _ 9 | _ _ 5 | _ _ 4 |
| _ _ 5 | _ 8 _ | _ 6 _ |
| _ _ _ | _ 6 1 | _ _ _ |
 ------- ------- -------
| _ _ _ | _ _ _ | _ 5 7 |
| 3 _ _ | _ _ _ | 8 _ _ |
| _ _ _ | 3 2 7 | _ _ _ |
 ------- ------- -------
```

Solving Sudoku - 03/02/21

```cpp
#define SUBSIZE 3
#define SIZE 9
int grid[SIZE][SIZE] = {0};

int main()
{
    int *board = import_board();
    display_board(board);
    cout << endl;

    if (solve_sudoku(board, 0))
    {
        display_board(board);
    }
    else
    {
        cout << "you're sudoku board has no solutions!" << endl;
    }
    return 0;
}
```

To solve the sudoku I implemented two functions. The first was a recursive function that would enable the search through the different branches of the sudoku.

When initially implementing this function I found that I had the issue that my condition to check if the sudoku was complete was in the wrong place so I moved this up from the bottom (where it was on my flowchart) to the top.

```
bool solve_sudoku(int *board, int boardPos)
{
    int row = boardPos / SIZE;
    int col = boardPos % SIZE;

    //is the sudoku complete
    if (row == SIZE)
    {
        return true;
    }

    //is it the end of the row
    if (col == SIZE)
    {
        row++;
        col = 0;
    }

    //is the value greater than zero
    if (board[boardPos] > 0)
    {
        if (solve_sudoku(board, boardPos + 1))
        {
            return true;
        }
    }
```

```
    //check if new value is valid
    for (int number = 1; number <= SIZE; number++)
    {
        if (valid_number(board, row, col, number))
        {
            board[boardPos] = number;
            if (solve_sudoku(board, boardPos + 1))
            {
                return true;
            }
        }
    }

    board[boardPos] = 0;
    return false;
}
```

The second function would be called from the solver to check for a valid number to fill the position in the board with.

```
bool valid_number(int *board, int row, int col, int num)
{
    //check if row is valid
    for (int i = 0; i < SIZE; i++)
    {
        if (board[(SIZE * row) + i] == num)
        {
            return false;
        }
    }

    //check if column is valid
    for (int i = 0; i < SIZE; i++)
    {
        if (board[(SIZE * i) + col] == num)
        {
            return false;
        }
    }

    //check if sub-board is valid
    int startRow = row - row % SUBSIZE;
    int startCol = col - col % SUBSIZE;
    for (int i = 0; i < SUBSIZE; i++)
    {
        for (int j = 0; j < SUBSIZE; j++)
        {
            if (board[(startRow + i) * SIZE + startCol + j] == num)
            {
                return false;
            }
        }
    }

    return true;
}
```

My first implementation of the sub-board check did not work. I had to change the code from the one below, to the final code

```
    //check if sub-board is valid
    int sub_row = row - row % subsize;
    int sub_col = col - col / subsize;
```

Once the changes, and some minor bug fixing was completed I was able to compile and run my program so that it was able to solve the sudoku.

```
PS G:\1204_Advanced_Programming\P1_Solving_Sudoku\3.3_Sudoku_solver>
.\sudoku.exe
 ------- ------- -------
| _ 2 _ | 8 _ _ | _ _ _ |
| 9 _ 1 | _ _ _ | _ _ _ |
| _ 7 _ | 2 4 6 | _ _ _ |
 ------- ------- -------
| 2 _ 9 | _ _ 5 | _ _ 4 |
| _ _ 5 | _ 8 _ | _ 6 _ |
| _ _ _ | _ 6 1 | _ _ _ |
 ------- ------- -------
| _ _ _ | _ _ _ | _ 5 7 |
| 3 _ _ | _ _ _ | 8 _ _ |
| _ _ _ | 3 2 7 | _ _ _ |
 ------- ------- -------


 ------- ------- -------
| 4 2 3 | 8 1 9 | 5 7 6 |
| 9 6 1 | 5 7 3 | 2 4 8 |
| 5 7 8 | 2 4 6 | 1 3 9 |
 ------- ------- -------
| 2 1 6 | 7 3 5 | 9 8 4 |
| 7 4 5 | 9 8 2 | 6 1 3 |
| 8 3 9 | 1 6 4 | 7 5 2 |
 ------- ------- -------
| 1 9 2 | 4 5 8 | 3 6 7 |
| 3 8 7 | 6 9 1 | 4 2 5 |
| 6 5 4 | 3 2 7 | 8 9 1 |
 ------- ------- -------
```

Optional Additional Work - 03/02/21

As my work was fully parameterised I was able to easily change it so that it would solve the sudoku for a 16x16 board.

```
#define SUBSIZE 4
#define SIZE 16
```

Unfortunately, in the time I was not able to modify the sudoku so that it was easily completable. However, I was able to successfully read and print the sudoku.

I found that the only necessary change to the working of the code was that the horizontal box lines were not parameterised.

```
PS G:\1204_Advanced_Programming\P1_Solving_Sudoku> .\su.exe
 ------- ------- -------
| _ 2 _ 8 | _ _ _ _ | _ _ _ _ | _ _ _ _ |
| 9 _ 1 _ | _ _ _ _ | _ _ _ _ | _ _ _ _ |
| _ 7 _ 2 | 4 6 _ _ | _ _ _ _ | _ _ _ _ |
| 2 _ 9 _ | _ 5 _ _ | 4 _ _ _ | _ _ _ _ |
 ------- ------- -------
| _ _ 5 _ | 8 _ _ 6 | _ _ _ _ | _ _ _ _ |
| _ _ _ _ | 6 1 _ _ | _ _ _ _ | _ _ _ _ |
| _ _ _ _ | _ _ _ 5 | 7 _ _ _ | _ _ _ _ |
| 3 _ _ _ | _ _ 8 _ | _ _ _ _ | _ _ _ _ |
 ------- ------- -------
| _ _ _ 3 | 2 7 _ _ | _ _ _ _ | _ _ _ _ |
| _ 2 _ 8 | _ _ _ _ | _ _ _ _ | _ _ _ _ |
| 9 _ 1 _ | _ _ _ _ | _ _ _ _ | _ _ _ _ |
| _ 7 _ 2 | 4 6 _ _ | _ _ _ _ | _ _ _ _ |
 ------- ------- -------
| 2 _ 9 _ | _ 5 _ _ | 4 _ _ _ | _ _ _ _ |
| _ _ 5 _ | 8 _ _ 6 | _ _ _ _ | _ _ _ _ |
| _ _ _ _ | 6 1 _ _ | _ _ _ _ | _ _ _ _ |
| _ _ _ _ | _ _ _ 5 | 7 _ _ _ | _ _ _ _ |
 ------- ------- -------
```

P2: Adders and Subtractors
https://stackoverflow.com/questions/6394741/can-a-c-function-return-more-than-one-value

Adders - 06/02/21

1) Write in your logbook codes to implement simple logic gates (AND, OR, XOR) as functions in C++.

When implementing the logic gates I know that each value is either a 1 or a zero so it can use the boolean data type. This allows me to use the logical operators.

```
bool func_and(bool A, bool B)
{
    return A && B;
}
bool func_or(bool A, bool B)
{
    return A || B;
}
bool func_xor(bool A, bool B)
{
    return A ^ B;
}
```

2) Describe, using suitable flowcharts or codes, how you might create a half-adder, using the functions you have written earlier.

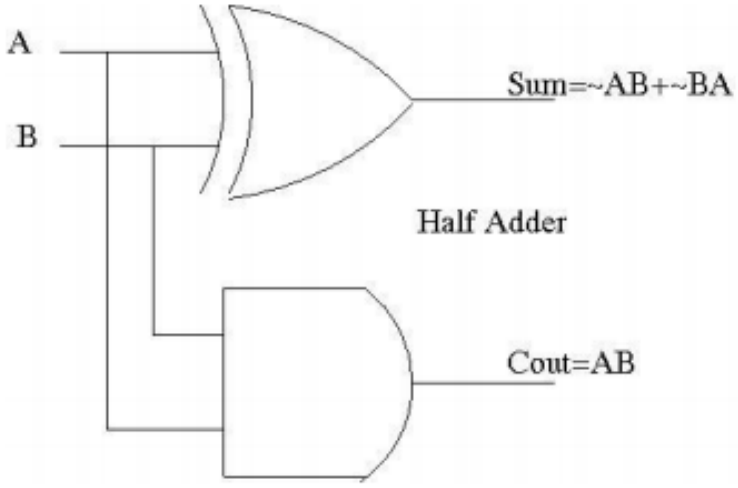


Figure 1: Half adder circuit

When implementing the half adder function I need to return more than one value. To do this I had to create a structure.

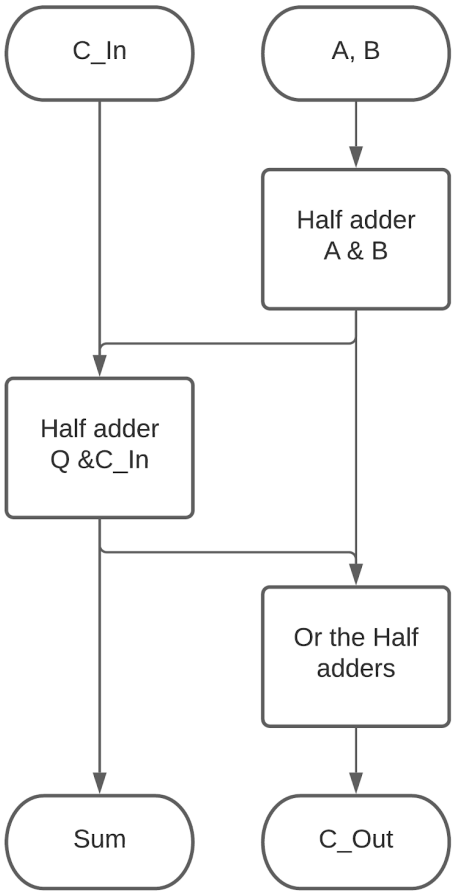
```
struct sum
{
    bool Q;
    bool c_out;
};
```

Then when creating the function, I simply called each logic gate for each of the values to be returned

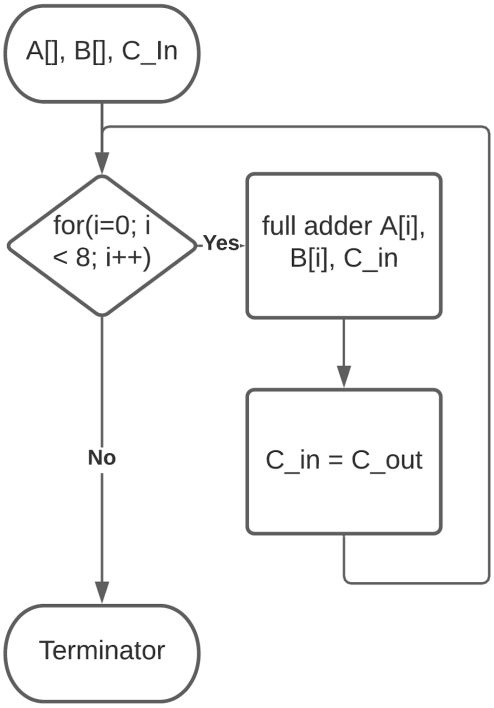
```
sum half_adder(bool A, bool B)
{
    sum result;
    result.Q = func_xor(A, B);
    result.c_out = func_and(A, B);

    return result;
}
```

3) Subsequently, draw a flowchart showing how you would implement a full-adder and an 8-bit adder with ripple propagation using full adders.



Full Adder



8-bit ripple counter

Implement single adder - 10/02/21

To implement the single adder I created a main program that would read in the values to use and output the calculated values.

```
int main()
{
    bool A;
    bool B;
    bool c_in;
    sum Q;

    cout << "Input your first binary number: ";
    cin >> A;
    cout << "Input your second binary number: ";
    cin >> B;
    cout << "Input your value for C_In: ";
    cin >> c_in;

    Q = full_adder(A, B, c_in);

    cout << endl << "Sum is: " << Q.Q << endl;
    cout << "Carry out is: " << Q.c_out;

    return 0;
}
```

To implement the full adder I used the flowchart that I prepared in the prep to write a function that would call the half adder function twice to create a full adder.

```
sum full_adder(bool A, bool B, bool c_in)
{
    sum fa;
    //half add A & B
    sum ha_1 = half_adder(A, B);
    //half add Q & c_in
    sum ha_2 = half_adder(ha_1.Q, c_in);

    //set the output
    fa.Q = ha_2.Q;
    fa.c_out = func_or(ha_1.c_out, ha_2.c_out);

    return fa;
}
```

Testing my implementation I found that I could correctly sum two single binary bits.

```
PS G:\1204_Advanced_Programming\P2_Adders_&_subtractors\3.1_Single_adder>
.\single.exe
Input your first binary number: 0
Input your second binary number: 0
Input your value for C_In: 1

Sum is: 1
Carry out is: 0
PS G:\1204_Advanced_Programming\P2_Adders_&_subtractors\3.1_Single_adder>
.\single.exe
Input your first binary number: 0
Input your second binary number: 1
Input your value for C_In: 1

Sum is: 0
Carry out is: 1
```

Implement 8-bit adder - 10/02/21

To implement the 8-bit adder I read in the ints of the two values. I then had to convert the value to an array of bools so my register function could access the information in binary. When printing the final number back out I had to implement the reverse function.

```
void conv_bin(int num, bool array[])
{
    for (int i = REG_SIZE - 1, j = 0; i >= 0; i--, j++)
    {
        if (num >= pow(2, i))
        {
            num -= pow(2, i);
            array[j] = 1;
        }
        else
        {
            array[j] = 0;
        }
        cout << array[j];
    }
}
int conv_dec(bool array[])
{
    int sum = 0;

    for(int i = REG_SIZE - 1, j = 0; i >= 0; i--, j++)
    {
        if (array[j] == 1)
        {
            sum += pow(2, i);
        }
    }

    return sum;
}
```

My register adder function followed the flowchart that I prepared in the prep. It would loop over the bits in the array completing the adder on the input registers and then update the carry in.

```
sum_arr reg_adder(bool A[], bool B[], bool c_in)
{
    sum_arr output;
    sum temp;

    //full adder each bit of the registers.
    for(int i = REG_SIZE - 1; i >= 0; i--)
    {
        temp = full_adder(A[i], B[i], c_in);
        output.Q[i] = temp.Q;
        c_in = temp.c_out;
    }

    //set c_out to notify of an overflow
    output.c_out = c_in;
    return output;
}
```

```
PS G:\1204_Advanced_Programming\P2_Adders_&_subtractors\3.2_8-bit> .\8
Input your first number (A): 5
Your corresponding value A in binary is: 00000101
Input your second number (B): 9
Your corresponding value B in binary is: 00001001
Input your value for C_In: 0

Sum is: 14
The corresponding value for sum in binary is: 00001110
Overflow bit is: 0
PS G:\1204_Advanced_Programming\P2_Adders_&_subtractors\3.2_8-bit> .\8
Input your first number (A): 250
Your corresponding value A in binary is: 11111010
Input your second number (B): 10
Your corresponding value B in binary is: 00001010
Input your value for C_In: 0

Sum is: 4
The corresponding value for sum in binary is: 00000100
Overflow bit is: 1
```

Subtraction - 10/02/21

As with the previous section I had to update my main function so that it could select the operation to perform.

```
    cout << "Select operation ( + or - ):";
    cin >> op;
    cout << num_1 << op << num_2 << endl;
    if ((int)op == 45)
    {
        num_2 *= -1;
        conv_bin(num_2, B);
    }
```

When converting two and from binary I had to implement a new function that would convert the array two and from two's complement. This would loop over the array "flipping the bits" then using the register adder it would add one.

```
void conv_two(bool array[])
{
    bool empty[REG_SIZE] = {0};

    for (int i = 0; i < REG_SIZE; i++)
    {
        if (array[i] == 1)
        {
            array[i] = 0;
        }
        else
        {
            array[i] = 1;
        }
    }

    sum_arr temp = reg_adder(array, empty, 1);
    for (int i = 0; i < REG_SIZE; i++)
    {
        array[i] = temp.Q[i];
    }
}
```

Testing the subtractor and two's complement implementation I found that my functions operated correctly.

```
PS G:\1204_Advanced_Programming\P2_Adders_&_subtractors\3.3_Subtraction>
.\sub.exe
Input your first number (A): -64
Your corresponding value A in binary is: 11000000
Input your second number (B): -16
Your corresponding value B in binary is: 11110000
Input your value for C_In: 0
Select operation ( + or - ):-
-64--16

Sum is: -48
Overflow bit is: 0
PS G:\1204_Advanced_Programming\P2_Adders_&_subtractors\3.3_Subtraction>
.\sub.exe
Input your first number (A): 80
Your corresponding value A in binary is: 01010000
Input your second number (B): -64
Your corresponding value B in binary is: 11000000
Input your value for C_In: 0
Select operation ( + or - ):+
80+-64

Sum is: 16
Overflow bit is: 1
```

P3: Into the imaginary realm
https://www.w3schools.com/cpp/cpp_classes.asp

Complex Number Class - 13/02/21

1) Design a class to represent the complex number with appropriate constructors. Your constructor should be overloaded appropriately with the correct overloading method. Implement your operators as member functions in your class.

To make my constructor I declared a function, with no type, that would then read in type of two floats. This was overloaded so that it would set the real and imaginary variables of the class to the values it read in.

```
class complex
{
public:
    complex(float re, float im)
    {
        real = re;
        imaginary = im;
    }
    float Re(void)
    {
        return real;
    }
    float Im(void)
    {
        return imaginary;
    }
    float mod(void)
    {
        return sqrt(pow(real, 2) + pow(imaginary, 2));
    }

private:
    float real;
    float imaginary;
};
```

2) Design appropriate overloading schemes for out-of-class overloading support for basic build-in operators.

I decided which operators I wanted to create overloading schemes for, namely the arithmetic and the comparative operators, and then overloaded them so that they would perform the operation on the class and return the appropriate value.

```
    complex operator+(complex &a) const
    {
        return complex(real + a.real, imaginary + a.imaginary);
    }
    complex operator-(complex &a) const
    complex operator*(complex &a) const
    complex operator/(complex &a) const

    void operator+=(complex &a)
    {
        real += a.real;
        imaginary += a.imaginary;
    }
    void operator-=(complex &a)
    void operator*=(complex &a)
    void operator/=(complex &a)

    bool operator==(complex &a) const
    {
        if ((real == a.real) && (imaginary == a.imaginary))
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
bool operator!=(complex &a) const
```

Impedance of circuits - 16/02/21

3) How do you find the real part of impedance in an RLC circuit?

$$Z = R + jX$$
$$\Re Z = R$$

4) How do you calculate the imaginary part of the impedance of an RLC circuit?

$$Z = R + jX$$
$$\Im Z = X$$
$$\Im Z = X_L - X_C$$
$$\Im Z = \omega L - \frac{1}{\omega C}$$

5) Write a simple function that takes in the total series resistance, capacitance, inductance and frequency of a series RLC circuit and convert it into impedance.

```
float RLC(float resistance, float inductance, float capacitance, float freq)
{
    float omega = 2 * M_PI * freq;

    float XL = omega * inductance;
    float XC = 1 / (omega * capacitance);

    return sqrt(pow(resistance, 2) + pow(XL - XC, 2));
}
```

6) Why is impedance important in a circuit?

Impedance is important in the calculator of the response of the circuit as ohm's law describes the voltage as:

$$V = IZ$$

And so to calculate the current of the RLC circuit it will be :

$$I = \frac{V}{Z}$$

Implement complex class - 17/02/21

To test the functions I created a main function that would initialise a complex number then print the result of the function to the terminal.

```
int main()
{
    complex A = complex(3, 4);

    std::cout << "Your value is: ";
    print_cartesian(A);

    std::cout << "The real part is: " << A.Re() << std::endl;
    std::cout << "The imaginary part is: " << A.Im() << std::endl;
    std::cout << std::endl;

    std::cout << "The modulus is: " << A.mod() << std::endl;
    std::cout << "The argument is: " << A.arg() << std::endl;
    std::cout << std::endl;

    std::cout << "The conjugate of your value is: ";
    print_cartesian(A.conj());
}
```

When the function returned a complex class I created a function that would correctly print the values to terminal. When more complex variables are needed to be printed this is useful as it reduces the amount of times the line has to be used.

```
void print_cartesian(complex a)
{
    if (a.Im() >= 0)
    {
        std::cout << a.Re() << " + j" << a.Im() << std::endl;
    }
    else
    {
        std::cout << a.Re() << " - j" << a.conj().Im() << std::endl;
    }
}
```

The resulting output to terminal for my main function was.

```
Your value is: 3 + j4
The real part is: 3
The imaginary part is: 4

The modulus is: 5
The argument is: 0.927295

The conjugate of your value is: 3 - j4
```

Convert RLC to impedance - 17/02/21

Due to misunderstanding the prep I had to modify my RLC function to return a complex class. To do this I changed the function type and modified the return function. This ultimately simplified my circuit.

For my main function I substituted the values from:
http://physicstasks.eu/1540/series-rlc-circuit

Into my RLC function to calculate the return value.

```
int main()
{
    float res = 50;
    float ind = 0.3;
    float cap = 0.000015;
    float freq = 50;

    complex imp = RLC(res, ind, cap, freq);

    std::cout << "Your complex impedance is: ";
    print_cartesian(imp);

    std::cout << "Your absolute impedance is: " << imp.mod();
}

complex RLC(float resistance, float inductance, float capacitance, float freq)
{
    float omega = 2 * M_PI * freq;

    float XL = omega * inductance;
    float XC = 1 / (omega * capacitance);

    return complex(resistance, XL - XC);
}
```

The output of the impedance calculated by my code was:

```
Your complex impedance is: 50 - j117.959
Your absolute impedance is: 128.118
```

RLC circuit simulator - 17/02/21

To calculate the current of the circuit I created a new RLC class. The classes constructor was based on my previous RLC function. I then had methods for returning values relating to the total impedance of the circuit and the complex current of the circuit.

```
class RLC
{
public:
    RLC(float resistance, float inductance, float capacitance, float volt,
float freq)
    {
        voltage = volt;
        omega = 2 * M_PI * freq;

        float XL = omega * inductance;
        float XC = 1 / (omega * capacitance);

        impedance = complex(resistance, XL - XC);
        current = complex(voltage, 0) / impedance;
    }

    complex imp(void)
    {
        return impedance;
    }
    complex cur()
    {
        return current;
    }

private:
    float voltage;
    float omega;
    complex impedance;
    complex current;
};
```

To create a user interface, I modified my main function to read in the values the values from the command line. It would then create the RLC class from this, hence calculating current.

```
int main()
{
    float res, ind, cap;
    float volt, freq;

    std::cout << "Your resistance is? ";
    std::cin >> res;
    std::cout << "Your inductance is? ";
    std::cin >> ind;
    std::cout << "Your capacitance is? ";
    std::cin >> cap;
    std::cout << "Your voltage amplitude is? ";
    std::cin >> volt;
    std::cout << "Your voltage frequency is? ";
    std::cin >> freq;

    RLC circuit = RLC(res, ind, cap, volt, freq);
    std::cout << std::endl << "Your current in complex form is: ";
    print_cartesian(circuit.cur());
    std::cout << "The magnitude of your current is: ";
    std::cout << circuit.cur().mod();
    std::cout << std::endl << "The phase difference between the voltage and
current is: ";
    std::cout << 0 - circuit.cur().arg();
}
```

The values I calculated are consistent with those on the linked site.

```
Your resistance is? 50
Your inductance is? 0.3
Your capacitance is? 0.000015
Your voltage amplitude is? 25
Your voltage frequency is? 50

Your current in complex form is: 0.0761532 + j0.179659
The magnitude of your current is: 0.195132
The phase difference between the voltage and current
is: -1.16988
```

In the series RLC cir

$$I_m = 0.2 \text{ A}.$$

The phase difference

$$\varphi = -67^\circ.$$

P4: Matrices and Vectors

Random Vector - 23/02/21

To create a vector of random double I had to use the random and vector libraries, this gave me the functions I required.

When initially initialising dre I used the default_random. However, due to issues with my MinGW installation this was deterministic. To fix this I instead initialised it with the time to get a true random string.

```
#include <random>
#include <ctime>
#include <vector>

int main()
{
    std::default_random_engine dre(std::time(NULL));
    std::uniform_real_distribution<double> dr(10, 20);

    std::vector<double> v;
    int vec_size;

    cout << "Enter the size of your vector." << endl;
    cin >> vec_size;
    for (int i = 0; i < vec_size; i++)
    {
        v.push_back(dr(dre));
    }

    for (int i = 0; i < vec_size; i++)
    {
        cout << v[i] << ", ";
    }
}
```

Matrix - 23/02/21

To create a matrix I used a vector of vectors. This allowed me to make it easily scalable. My implementation means it has to be a square matrix but as this is intended for a tridiagonal matrix this is not an issue.

To correctly initialise the matrix I had to change the code. I used a helpful suggestion I found on stackoverflow to initialise it.

```
    int mat_size;
    cout << "Enter the size of your Matrix." << endl;
    cin >> mat_size;

    std::vector<std::vector<double>> M(mat_size); //Luchian Grigore at
https://stackoverflow.com/questions/12375591/vector-of-vectors-to-create-matrix
    for (int i = 0; i < mat_size; i++)
    {
        M[i].resize(mat_size);
    }

    for (int i = 0; i < mat_size; i++)
    {
        for (int j = 0; j < mat_size; j++)
        {
            M[i][j] = dr(dre);
        }
    }

    for (int i = 0; i < mat_size; i++)
    {
        for (int j = 0; j < mat_size; j++)
        {
            cout << M[i][j] << " ";
        }
        cout << endl;
    }
```

Linear Equation Solver - 23/02/21

To convert the function from c to c++ I converted all the the pointers to vectors. I also found it useful to change the naming convention to something that is more intuitive.

```
std::vector<double> TridiagonalSolve(double c1, std::vector<double> &diag,
std::vector<double> &vect, int n)
{
    int i;
    std::vector<double> off_diag(n);
    std::vector<double> output(n);
    double id;

    // Set the off diagonal elements
    for (i = 0; i < n; i++)
    {
        off_diag[i] = c1;
    }

    //forward bit
    off_diag[0] /= diag[0]; //if /0 rearrange equations
    vect[0] /= diag[0];

    for (i = 1; i < n; i++)
    {
        id = diag[i] - off_diag[i - 1] * c1;
        off_diag[i] /= id; // Last value calculated is redundant.
        vect[i] = (vect[i] - vect[i - 1] * c1) / id;
    }

    // Now back substitute.
    output[n - 1] = vect[n - 1];

    for (i = n - 2; i >= 0; i--)
    {
        output[i] = vect[i] - off_diag[i] * output[i + 1];
    }

    return output;
}
```

Tridiagonal Solver - 24/02/21

To test the implementation of the function I created a main program that would create a tridiagonal matrix and a vector. Find the product. Then using the matrix and the product it finds the original vector again.

```
int main()
{
    //determine size of the matrix and vector
    int size;
    cout << "Enter the size of your Matrix." << endl;
    cin >> size;

    //generate tridiagonal matrix and vector
    std::vector<double> vec = VectorGenerate(size);
    std::vector<std::vector<double>> tridiag = TridiagonalGenerate(size);

    cout << endl
         << "Your tridiagonal matrix is:" << endl;
    TridiagonalPrint(size, tridiag);

    cout << "Your vector is:" << endl;
    VectorPrint(size, vec);

    //multiply tridiagonal matrix and vector
    std::vector<double> product = multiplication(size, vec, tridiag);

    cout << endl
         << "The product vector is:" << endl;
    VectorPrint(size, product);

    //solve for original vector from matrix and product
    std::vector<double> diag;
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            if (i == j)
            {
                diag.push_back(tridiag[i][j]);
```

```
            }
        }
    }

    std::vector<double> original = TridiagonalSolve(tridiag[0][1], diag,
product, size);

    cout << endl
         << "The original vector was:" << endl;
    VectorPrint(size, original);
}
```

The result on the command line shows that it is correctly calculating the original vector.

```
Enter the size of your Matrix.
6

Your tridiagonal matrix is:
10.7061 17.1776       0       0       0       0
17.1776 13.0758 17.1776       0       0       0
      0 17.1776 18.2992 17.1776       0       0
      0       0 17.1776 10.7528 17.1776       0
      0       0       0 17.1776 10.3502 17.1776
      0       0       0       0 17.1776 10.0471

Your vector is:
19.1133 17.4045 15.8379 16.5649 11.1185   13.23

The product vector is:
503.594 827.953 873.334 641.165 626.884 323.911

The original vector was:
19.1133 17.4045 15.8379 16.5649 11.1185   13.23
```

VectorGenerate() was taken from my main function on page 32.
TridiagonalGenerate() was a modified version of the main function on page 33. Modified so it would only fill the diagonal.

```
    double rand = dr(dre);
    for (int i = 0; i < mat_size; i++)
    {
        for (int j = 0; j < mat_size; j++)
        {
            if (i - 1 <= j && j <= i + 1)
            {
                mat[i][j] = rand;
            }
            else
            {
                mat[i][j] = 0;
            }
            if (i == j)
            {
                mat[i][j] = dr(dre);
            }
        }
    }
```

VectorPrint() and TridiagonalPrint() were also taken from the main functions.
The multiplication was also handled by its own function. It would take in the reference to the matrix and vector and create a vector product.

```
std::vector<double> multiplication(int size, std::vector<double> &vec,
std::vector<std::vector<double>> &tridiag)
{
    std::vector<double> product;
    for (int i = 0; i < size; i++)
    {
        long double sum = 0;
        for (int j = 0; j < size; j++)
        {
            sum += tridiag[i][j] * vec[j];
        }

        product.push_back(sum);
    }
            return product;
}
```

Complex Matrices - 24/02/21

To implement my complex class from the previous lab I included a header file with the definition for the complex class. I also ported across the code to print the matrix in cartesian form.

```
void print_cartesian(complex a)
{
    if (a.Im() >= 0)
    {
        std::cout << std::setw(7) << a.Re() << " + j" << std::setw(7) << a.Im();
    }
    else
    {
        std::cout << std::setw(7) << a.Re() << " - j" << std::setw(7) <<
a.conj().Im();
    }
}
```

I then found and replaced every double data type and changed it to the complex data type. To generate a random complex number I created a function that would return the constructor with two random doubles.

```
complex ComplexRandom(void)
{
    return complex(dr(dre), dr(dre));
}
```

The outputfor this worked, displaying it in complex form. However, it is not particularly intuitive to follow so if I had more time on this lab I would create a morem user friendly interface.

```
Enter the size of your Matrix.
4

Your tridiagonal matrix is:
15.9028 + j11.9031 11.3228 + j19.0814         0 + j        0         0 + j        0
11.3228 + j19.0814 16.3886 + j18.9199 11.3228 + j19.0814         0 + j        0
        0 + j        0 11.3228 + j19.0814   18.512 + j13.5472 11.3228 + j19.0814
        0 + j        0         0 + j        0 11.3228 + j19.0814 11.3188 + j12.0052

Your vector is:
16.8854 + j12.0728 18.0009 + j13.3699 17.6107 + j11.6462 10.8323 + j12.9583

The product vector is:
73.5263 + j887.848 -19.9459 + j1486.49 -7.6713 + j1302.46 -55.7826 + j744.621

The original vector was:
16.8854 + j12.0728 18.0009 + j13.3699 17.6107 + j11.6462 10.8323 + j12.9583
```