V.1

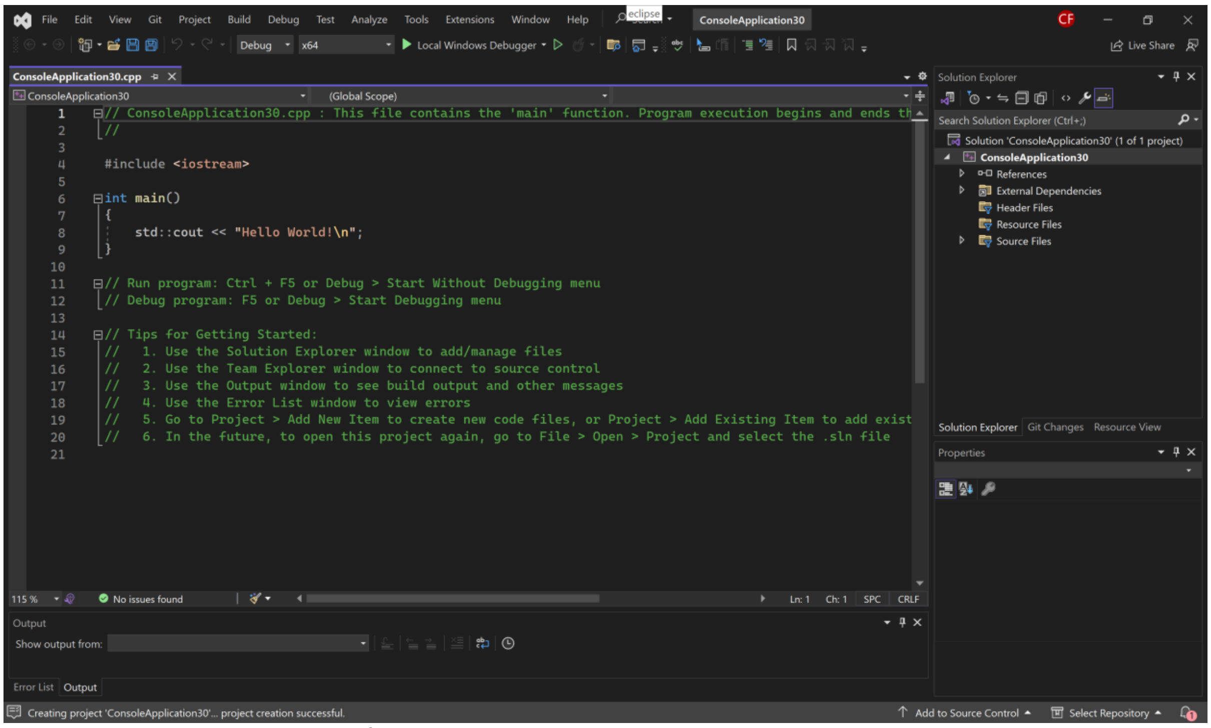
Software Engineering Taster Session

C++ is a general purpose programming language with the ability to write object-oriented programs and allows you to manipulate memory directly. It is the main language for the Software Engineering and Games Development course. It is one of the most widely used programming languages in industry and allows you to write fast and efficient code.

Ever used Google? It's powered by C++ code, as well as lots of the CGI used in film.

- Your PC should be logged in and Visual Studio 2022 should be opened, if it isn't please ask the tutor to log you in.
- Under "Create a New Project" select "C++ Console App".
- Give your project a name. You can leave the default location for where it will be saved.
- Select "Create".

V.1



```
// ConsoleApplication30.cpp : This file contains the 'main' function. Program execution begins and ends th
//

#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}

// Run program: Ctrl + F5 or Debug > Start Without Debugging menu
// Debug program: F5 or Debug > Start Debugging menu

// Tips for Getting Started:
//   1. Use the Solution Explorer window to add/manage files
//   2. Use the Team Explorer window to connect to source control
//   3. Use the Output window to see build output and other messages
//   4. Use the Error List window to view errors
//   5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add exist
//   6. In the future, to open this project again, go to File > Open > Project and select the .sln file
```

Figure 1. Visual Studio 2022 Interface

Technical Specification

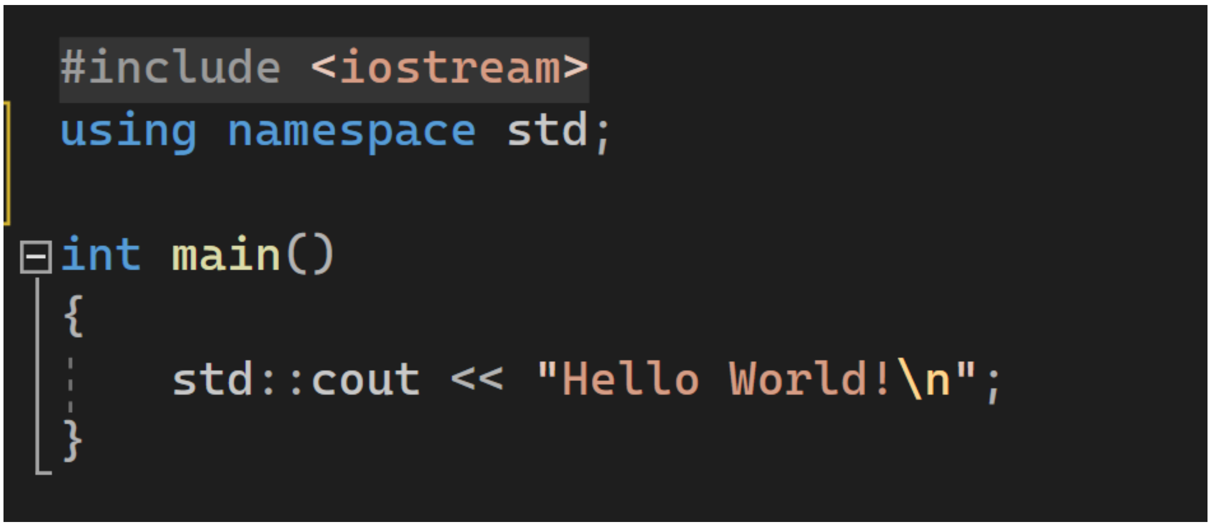
We are to build a guessing game where the user should guess a number between 1 and 10.
Additional features we want to add include

- To remind the user which numbers they have already guessed (in ascending order),
- whether they guessed the same number again,
- and once they have guessed the correct number, how many guesses it took to guess the right number.

You will learn about various programming constructs: statements, loops, conditions, variables, etc.

Step 1: The Foundations

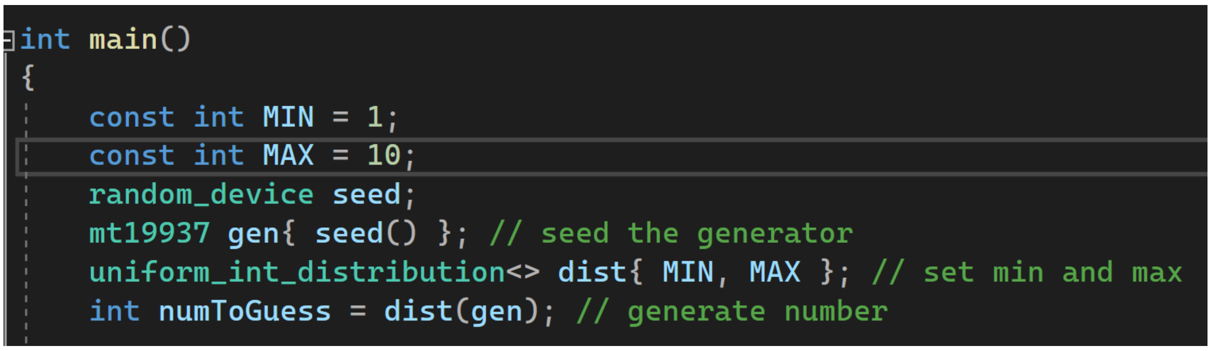
- You will need to add `using namespace std;` below the `#include <iostream>`This allows you use the read-in, output, string, and other *standard* functionalities.



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

int main()
{
    std::cout << "Hello World!\n";
}
```

Figure 2. Adding using namespace std;

- The first bit of coding you will need to do is to generate a random number for the user to guess.
- Generating a truly random number remains an unsolved problem in computing, however, we can generate a number which comes close enough to random for this exercise.
- We also need a min and max value for the random number to generate. To store these as a constants, will make it easier to change them later on in case you change you mind about the range. If you use something called "magic numbers" – hard coded numbers in your code – it makes it much more difficult to change the range quickly later on.
- You will also need to include another library to use the functions for creating a random number. The library is called `#include <random>`. Just add it at the top with the other one.
- Here is the code for generating a random number. It is not very important to understand what exactly it is doing for now.

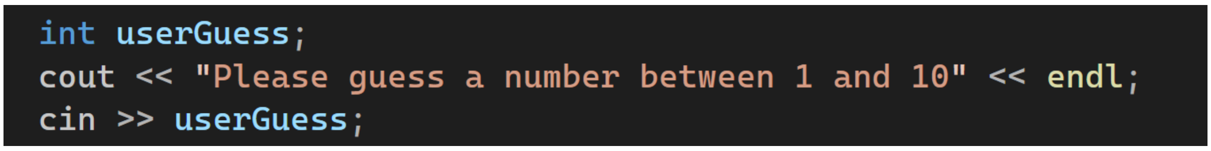


```
int main()
{
    const int MIN = 1;
    const int MAX = 10;
    random_device seed;
    mt19937 gen{ seed() }; // seed the generator
    uniform_int_distribution<> dist{ MIN, MAX }; // set min and max
    int numToGuess = dist(gen); // generate number
```

Figure 3. Generating random number

Step 2: Asking the user for input

- We need to ask the user to input a number between 1 and 10 (using `cout <<`). You might want to add an `<< endl` statement at the end of your output statement, which creates a line break and a carriage return.
- We also need to read in the user input (using `cin >>`). We're storing the number in an `int` variable, which is integer.

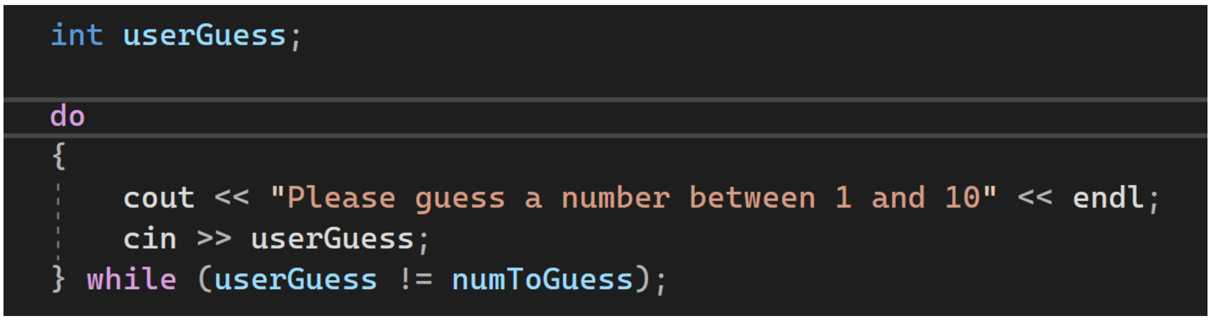


```cpp
int userGuess;
cout << "Please guess a number between 1 and 10" << endl;
cin >> userGuess;
```

Figure 4: Input and output.

Step 3: Asking the user to repeat until number has been guessed

- We want to repeat the same instructions until the user has guessed the number. We don't know how many guesses the user will need, so we need to wrap the code into a loop.
- We're using a **do-while** loop as the code needs to run **at least one time**.
- The code will continue looping as long as the condition is met – in this case while the user has **not** guessed the number.
- Also, we need to move the introduction of our variable outside of the loop as variables only exist within the scope they've been created in, in this case it would have been the loop, however we also need it to check our condition, so we need to move the declaration to outside of the loop.

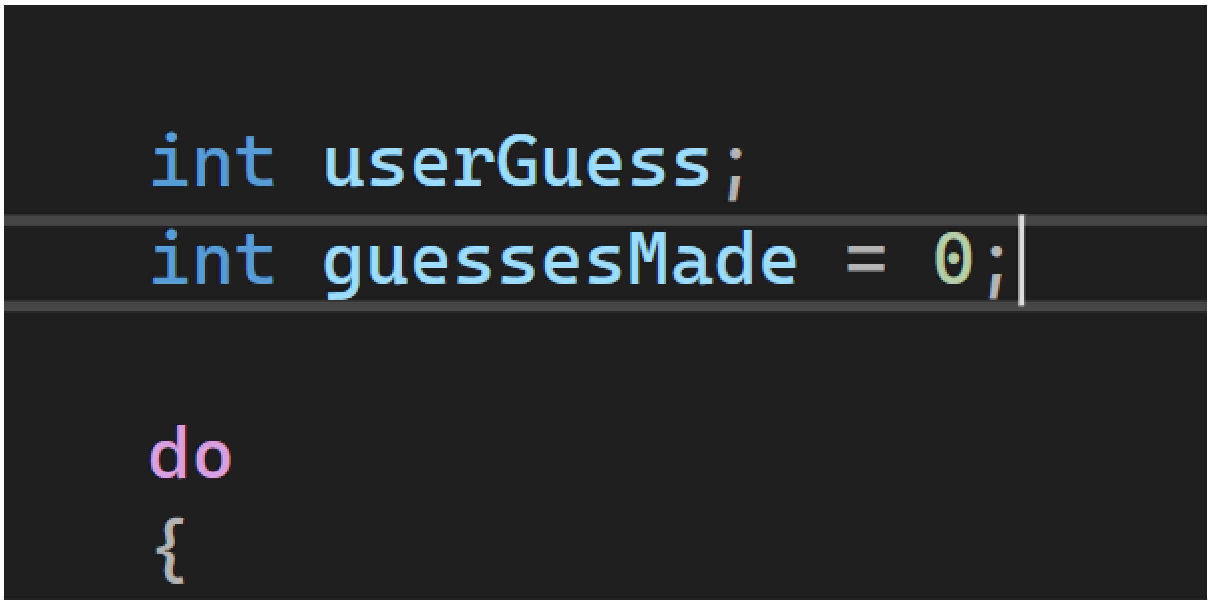


```cpp
int userGuess;

do
{
    cout << "Please guess a number between 1 and 10" << endl;
    cin >> userGuess;
} while (userGuess != numToGuess);
```

Figure 5. Looping instructions.

- Maybe you want to tell the user once they have guessed the correct number, so we should add an output statement underneath our loop.

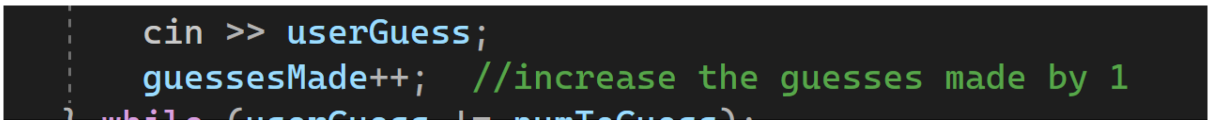
- I also want to let the user know how many guesses it took them to guess the right number.
- We will need to introduce another integer variable which increases every time the user made a guess.
- We want to introduce the variable in the same place as the input variable and we want to give it the initial value of 0.



```
int userGuess;
int guessesMade = 0;

do
{
```

Figure 5: Figure 6. Introducing counting variable.

- Using the **++** operator, we increase the value of the variable by 1 every time the user made a guess.

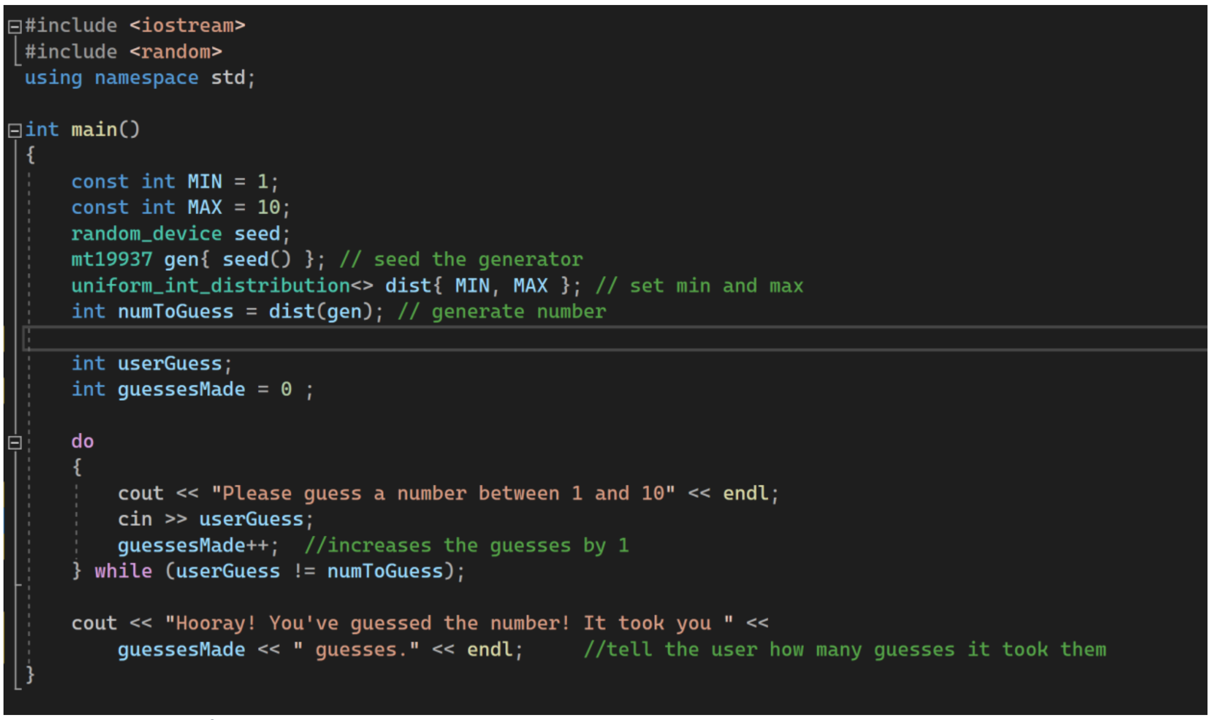


```
    cin >> userGuess;
    guessesMade++;   //increase the guesses made by 1
} while (userGuess != numToGuess);
```

Figure 6: Increasing counting variable

- So far your code should look something like this:

V.1



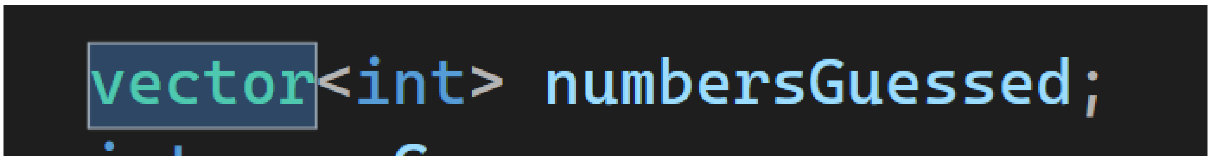
```
#include <iostream>
#include <random>
using namespace std;

int main()
{
    const int MIN = 1;
    const int MAX = 10;
    random_device seed;
    mt19937 gen{ seed() }; // seed the generator
    uniform_int_distribution<> dist{ MIN, MAX }; // set min and max
    int numToGuess = dist(gen); // generate number

    int userGuess;
    int guessesMade = 0 ;

    do
    {
        cout << "Please guess a number between 1 and 10" << endl;
        cin >> userGuess;
        guessesMade++;  //increases the guesses by 1
    } while (userGuess != numToGuess);

    cout << "Hooray! You've guessed the number! It took you " <<
        guessesMade << " guesses." << endl;     //tell the user how many guesses it took them
}
```

Figure 7: Code so far.

- Run and test your code – play the game and see what happens.
- Congrats! You've coded the basic functionality of a guessing game, but…

Step 4: Let's make it more interesting – telling the user which numbers they have guessed

- What if the user has a terrible short-term memory and keeps guessing the same number? One thing we could do is to tell them that they have already guessed the number, but we can go even further and output all the numbers the user has guessed so far before they try again! We should do both!
- We will need some sort of container to store the previously guessed integers. A `vector` gives us the flexibility to store as many values as we want.
- Declare the vector just above the variable for user input. The syntax looks like this:



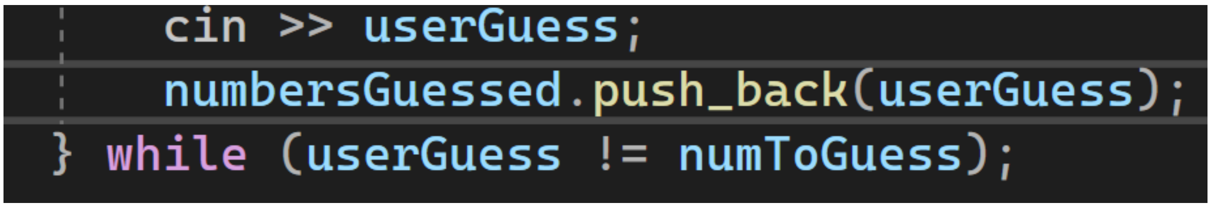
```
vector<int> numbersGuessed;
```

Figure 8: Syntax vector

- We want to add a new value to it every time the user has guessed the number.
- To be able to do this, we need to use a certain functionality (something called a **function**) of the vector, which pushes the value to the end. Vectors only allow us to

push data to the back. In our case, it does not matter as we want to sort the numbers anyway.

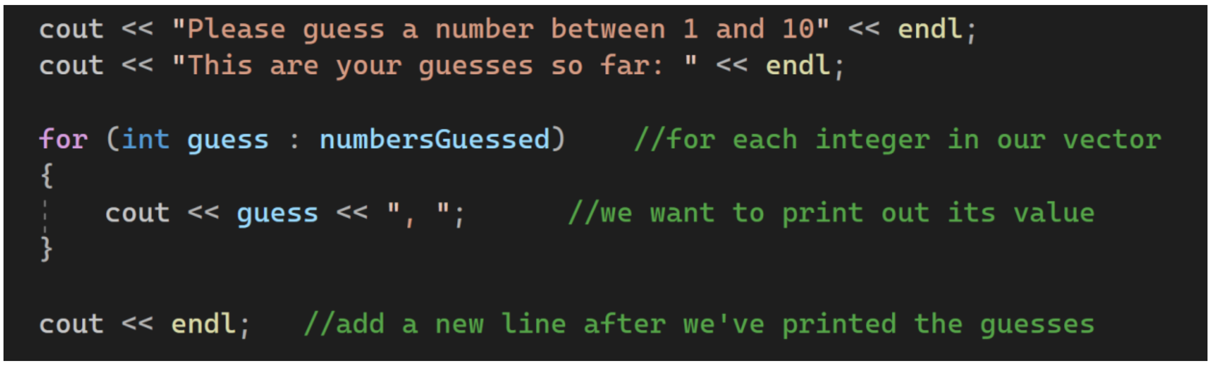
- Add the function to the bottom of the **body** (anything between the two brackets of the loop is the body) of your loop.



```
    cin >> userGuess;
    numbersGuessed.push_back(userGuess);
} while (userGuess != numToGuess);
```

Figure 9: Adding data to vector

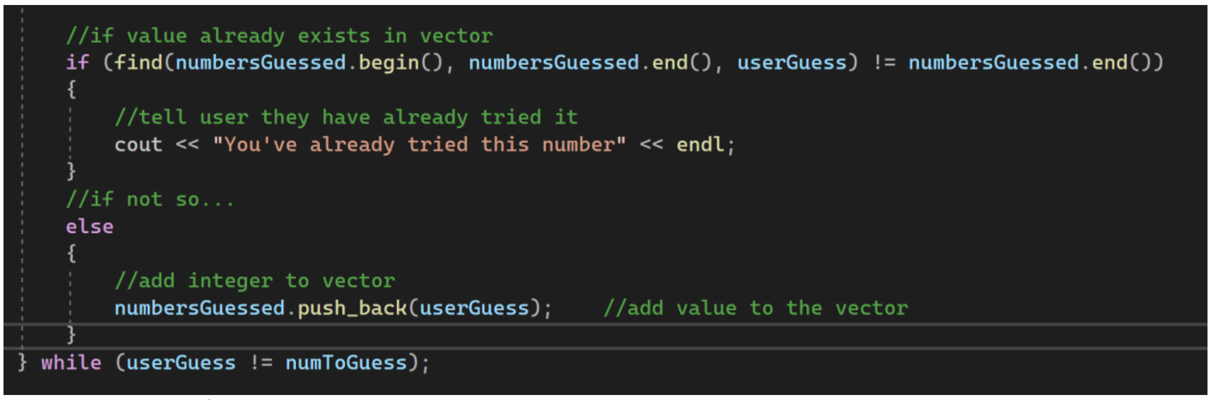
- So, now we can tell the user which numbers they have guessed every time before they guess again!
- We will do this by using a range-based for loop (some of you might know it as a for-each loop). The second name gives away how the loop works. For each item in our vector (in this case they are integers), we want to do something. In our little game, for each item, we want to print their value out.
- Here's how you can do this:



```
cout << "Please guess a number between 1 and 10" << endl;
cout << "This are your guesses so far: " << endl;

for (int guess : numbersGuessed)    //for each integer in our vector
{
    cout << guess << ", ";        //we want to print out its value
}

cout << endl;   //add a new line after we've printed the guesses
```

Figure 10: Output all numbers which have been guessed so far

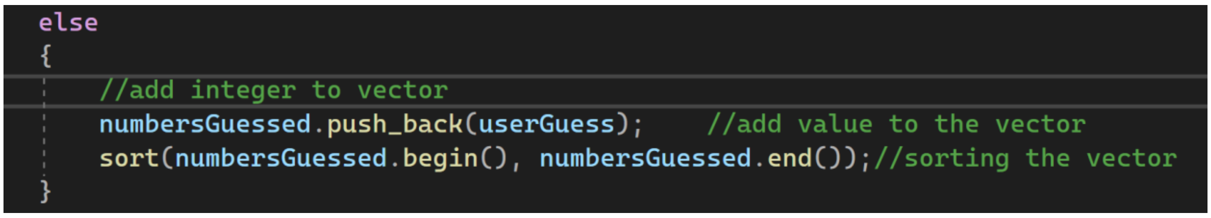
- You will have noticed that we needed to introduce a new variable in the loop. This variable represents the current integer we are working with. This variable has a new value every iteration of the loop.
- I have also added a new line after the loop has ended, so the next input or output will be in a new line.
- We still want to tell the user if they've already guessed the number.
- Instead of looping again through our vector to compare every value with the user input, which is possible, we can make use of an existing **function**, which does this for us.
- **If** this function finds out this value already exists in our vector, we want to tell the user that they've already tried this number, **else** we want to add the number to our vector.
- We are going to make use of a conditional statement (called and **if-else** statement) to do this.



```
    //if value already exists in vector
    if (find(numbersGuessed.begin(), numbersGuessed.end(), userGuess) != numbersGuessed.end())
    {
        //tell user they have already tried it
        cout << "You've already tried this number" << endl;
    }
    //if not so...
    else
    {
        //add integer to vector
        numbersGuessed.push_back(userGuess);    //add value to the vector
    }
} while (userGuess != numToGuess);
```

Figure 11: Check if the number already exists

- What this `find` function does, is to check from the beginning to the end of the vector if the given value (the user input) already exists, if so it will return **true**, which executes the code in our **if**-body. If it does not find it, it will return **false**, which ignores the **if**-body and executes the **else**-body.
- Now we want to sort our vector, so the user can see which numbers they have guessed in an ascending order.
- There is also a simple function provided to sort the whole vector, which we will make use of. In your first year on the Software Engineering course, you will be expected to write your own simple sorting algorithm, so you will learn how these kind of algorithms work eventually.

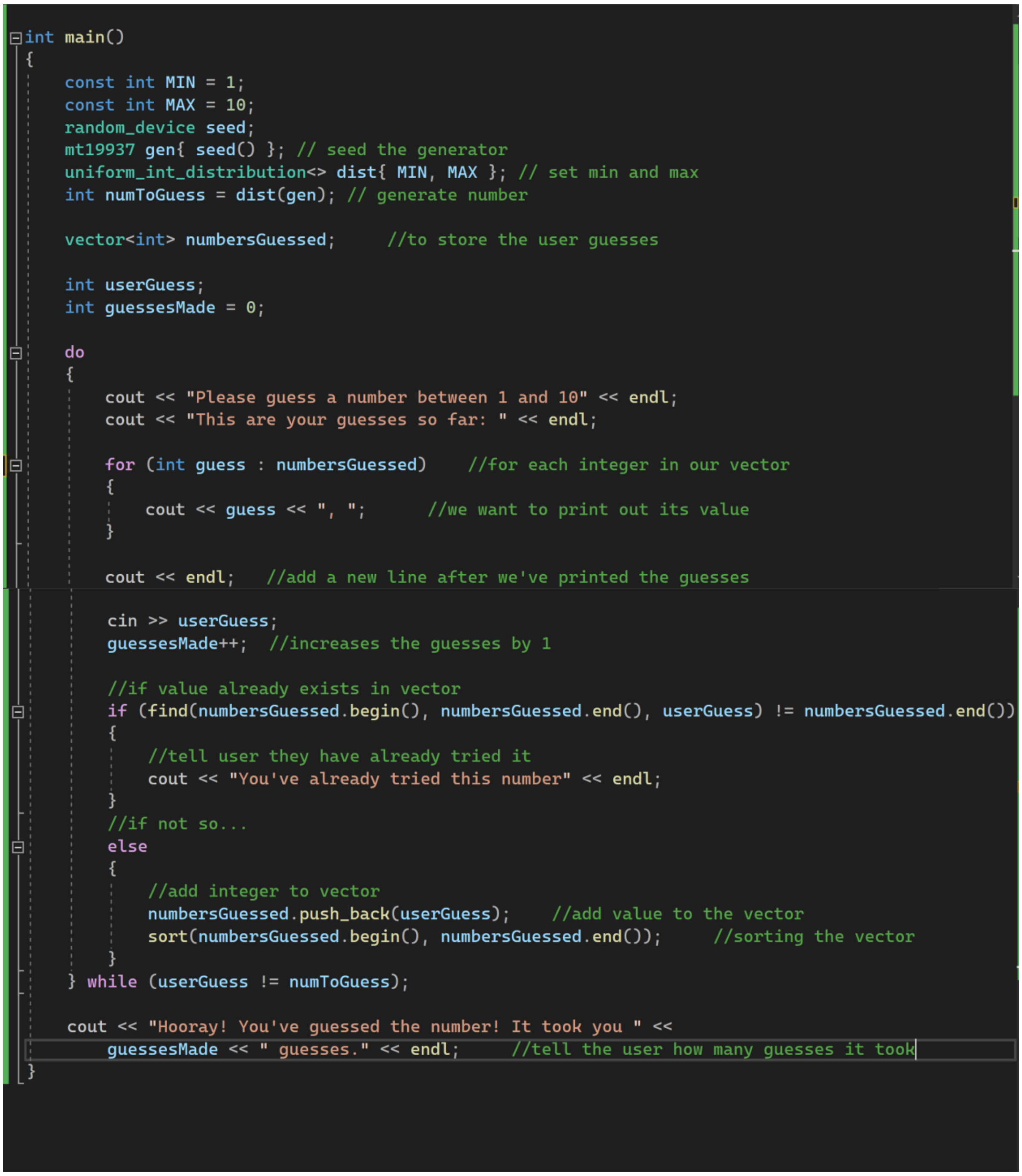


```
else
{
    //add integer to vector
    numbersGuessed.push_back(userGuess);    //add value to the vector
    sort(numbersGuessed.begin(), numbersGuessed.end());//sorting the vector
}
```

Figure 12: Sorting function

- The `sort` function sorts the vector from the beginning to the end.
- You code should look something like this now:

V.1



```
int main()
{
    const int MIN = 1;
    const int MAX = 10;
    random_device seed;
    mt19937 gen{ seed() }; // seed the generator
    uniform_int_distribution<> dist{ MIN, MAX }; // set min and max
    int numToGuess = dist(gen); // generate number

    vector<int> numbersGuessed;     //to store the user guesses

    int userGuess;
    int guessesMade = 0;

    do
    {
        cout << "Please guess a number between 1 and 10" << endl;
        cout << "This are your guesses so far: " << endl;

        for (int guess : numbersGuessed)    //for each integer in our vector
        {
            cout << guess << ", ";      //we want to print out its value
        }

        cout << endl;   //add a new line after we've printed the guesses

        cin >> userGuess;
        guessesMade++;  //increases the guesses by 1

        //if value already exists in vector
        if (find(numbersGuessed.begin(), numbersGuessed.end(), userGuess) != numbersGuessed.end())
        {
            //tell user they have already tried it
            cout << "You've already tried this number" << endl;
        }
        //if not so...
        else
        {
            //add integer to vector
            numbersGuessed.push_back(userGuess);    //add value to the vector
            sort(numbersGuessed.begin(), numbersGuessed.end());     //sorting the vector
        }
    } while (userGuess != numToGuess);

    cout << "Hooray! You've guessed the number! It took you " <<
        guessesMade << " guesses." << endl;     //tell the user how many guesses it took
}
```

Figure 13: Final program

Congratulations! You've reached the end of today's short exercise. Now you know the basics of programming, can you think of any additional features you might like to implement in order to improve our little guessing game?
Here are some suggestions:

- Tell the user if the guess was lower or higher than the number to guess
- Introduce the notion of "lives" → if the user doesn't get it within 5 guesses, the computer wins

V.1

- Add rounds → if the user wins, they get another round and the range of number increases

Some advanced suggestions:

- Introduce even more specific feedback, whether they're "hot" or "cold" → close or far from the number to guess
- The user could score points depending on how often they needed to guess (alternatively the user can score points each guess depending on how close they are)
- They could get an extra life in the harder rounds

Super advanced suggestions:

- Add a high score → save a high score to a text file and read it in each time you run the game again