```
//============================
// Section 9 Controlling Program Flow
//============================

// Controlling program flow
Sequence
- Ordering statements sequentially

Selection
- Making decisions

Iteration
- Looping or repeating

// Main statements
if statement
if-else statement
Nested if statements
switch statement
conditional operator ?: (ternary)

// Iteration
for loop
range-based
for loop
while loop
do-while loop
continue and break
infinite loop
nested loop

// The if statement
if (expression){Do something};
- if the expression is a boolean true, do the things
- if the expression is false, then skip the statement

If the statement to execute is a single line, you don't need the braces
The braces make it a 'block' statement

// The if/else statement
if (expression){
    // If expression is true do this
} else {
    // else if the expression is false do this
}

// The if/else if statement
if(expression){
    // if true do this
} else if (expression){
    // else if this is true do this
} else {
    // Default
}

// Nested if statements
if(expression){
    if (expression){
    }
}
```

```
// Dangling else problem - which if does an else belong to?
if (expression)
    if (expression)
        // do something
    else
        // do something else

// In C++ the else belongs to the closest if statement
// use block statements to avoid this problem entirely

// Nicely formatting dollars
#include <iomanip>
setprecision(2); // Number of decimal points it will print

// Switch\case statement
// must evaluate to an integral or enumeration type - constants or literals known as compile time
// Numbers will work, character literals will also work


switch (integerControlExpression){
    case expression1:
    //statements
    break;

    case expression2:
    //statements
    break;

    default:
}

If you omit the break statement the cases will fall through

// With an enumeration type
enum Color{
    red,green,blue
};
Color screenColor{};

switch(screenColor){
    case red:
        //whatever
    case green:
        {
            const int whatever{0};
            break;
        }

    case blue:
        //whatever
        break;
    default:
        //whatever
}

// Switch statement facts
- Constrol expression must evaluate to an integer type
- Case expressions must be constant expressions that evaluate to integers or integer literals
- Once a match occurs all following case statements are executed until a break is encountered
- Provide a break statement for each case
- default is optional but should be handled
- note that if you need a variable in your case statements you need to create a block {}
```

```
// Conditional operator
(conditional expression) ? expression1 : expression2

The conditional expression must be a boolean value
If it's true, the first expression is returned.
If it's false, the second is returned
Similar to if\else
Ternary operator
Useful when used inline
very easy to abuse!

result = (a > b) ? a : b;

// Looping
- Third basic block of programming
- sequence, selection, iteration
Iteration or repetition
Allows the execution of a statement or block of statements repeatedly
Loops are made up of a loop condition and the body which contains the statements to repeat

 // Use cases
 Execute a loop:
 - specific number of times
 - for each element in a loop
 - while a condition is true
 - until a condition becomes false
 - until we reach the end of some input stream
 - forever
 - many, many more

// Looping structures
for loop
- Iterate a specific number of times

wange-based for loop
- one iteration for each element in a range or collection

while loop
- while a condition is true
- stops when it becomes false
- condition checked at the beginning of each iteration

do-while loop
- iterate while a condition is true
- stop when the condition becomes false
- checks at the end of every iteration

// The for loop
for (initialization statement; condition to test; increment){
    //statements
}

(can also be written without the braces but only a single line)

for (int i{0}; i <= 10, i++){
    std::cout << someArray[i] << endl;
}
NOTE: When we initialize the i inside the loop, it's only available inside the loop
NOTE: be careful of bounds checking - if you start at

// Compound loops
for(int i{}, int j{}; i < 10, j < 10; i++, j++){
    // This is a really ugly syntax
}

NOTE: You may get a warning if you don't specify unsigned for the iterator and
you use something like the size() method on a vector which does use unsigned by default
(no negative numbers);
```

```
// Range-based for loop
for (dataType variableName: sequence){
    //statements
}

for (int items: itemArray){ // Use the same data type as the array
    std::cout << items << std::endl;
}

for (auto items: itemArray){ // compiler will automatically decide
    std::cout << items << std::endl;
}

// It's possible to use it with an inline collection
for(auto temp: {65,67,78,98}){
}

// You can also use it to iterate over a string
for (auto letters:"Josie"){

}

// More on <iomanip>
#include <iomanip>
cout << fixed << setprecision(1); // will set to 1 decimal

// While loop - this loop might never execute depending on the test condition
while (expression){
    //statements
}

- You can create an infinite loop by doing:
while(true){
    // This loop will never end unless you break out
}

- commonly used with boolean flags

// do while loop
do {
    statements;
} while (expression)

This will execute at least once since we check at the end

// Continue and break
continue
- no further statements in the body of the loop are executed
- control immediately goes back to the beginning of the loop for the next iteration

break
- no further statements in the body of the loop are executed
- loop is terminated
- control immediately goes to the statement following the loop

// Infinite loops
Loops whose condition always evaluates to true
Usually it's a mistake
Sometimes programmers use them and include break statements to control them
Sometimes they are exactly what we need
- Event loop in an event-driven program
- Operating system


// Infinite for loop
for(;;){}

// Infinite while
while(true){}
```

```cpp
// Infinite do while
do{}while(true);

// Example
while(true){
    char again{};
    cout << "Do you want to loop again?";
    cin >> again;
    if(again == "N" || again == 'n'){
        break;
    }
}

// Nested loops
Loop nested within another loop
Can be as many levels deep as the program needs
Very useful with multi-dimensional data structures
Outer loops vs inner loops - one iteration for outer then all the inner loops

#include <iostream>
int main(){
    std::cout << "Multiplication tables below:" << std::endl;
    std::cout << "============================" << std::endl;
    for (int i{ 1 }; i <= 10; i++) {
        for (int j{ 1 }; j <= 10; j++) {
            std::cout << i << " * " << j << " = " << i * j << std::endl;
        }
        std::cout << "============================" << std::endl;
    }
}

// Example iteration over a grid
int grid[5][3]{};

for (int row{0}; row < 6; row++){
    for (int col{0}; col < 3; col++){
        grid[row][col] = 1000;
    }
}

//Example with vectors
vector<vector<int>> twoDimensionalVector{
    {1,2,3},
    {10,20,30,40},
    {100,200,300,400,500}
};

for (auto row:twoDimensionalVector){
    for(auto col:twoDimensionalVector){
        std::cout << col << " ";
    }
    std::cout << endl;
}

//Example with vectors - this way we can use different sizes for the rows
std::vector<std::vector<int>> twoDimensionalVector{
    {1,2,3},
    {10,20,30,40},
    {100,200,300,400,500}
};

for (auto row : twoDimensionalVector) {
    for (auto col : row) { // In this loop we need to iterate over the previous loop
        std::cout << col << " ";
    }
    std::cout << std::endl;
}
```

```cpp
// Histogram example
#include <iostream>
#include <vector>
#include <string>

int main(){

    int numberOfItems{};
    std::cout << "How many data items do you have? ";
    std::cin >> numberOfItems;

    std::vector<int> data{};
    for (int i{ 1 }; i <= numberOfItems; i++) {
        int dataItem{};
        std::cout << "Please enter the data to push into the vector: ";
        std::cin >> dataItem;
        data.push_back(dataItem);
    }

    std::cout << "Displaying histogram:" << std::endl;

    for (auto item : data) {
        for (int i{ 0 }; i < item; i++) {
            if (i % 5 == 0) { // print a * for every 5 items
                std::cout << "*";
            } else {
                std::cout << "-";
            }
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
    return 0;
}

//============================
// Section 9 Controlling Program Flow Challenge
//============================

#include <iostream>
#include <vector>
#include <string>

void clearScreen() {
    // This translates to a code that clears the console
    std::cout << "\033[2J\033[1;1H";
}

int main(){

    char selection{};
    std::vector<double> numbersList{};
    std::string message{};
    do {
        std::cout << "Please make a selection: " << std::endl;
        std::cout << "P - Print numbers" << std::endl;
        std::cout << "A - Add a number" << std::endl;
        std::cout << "M - Display the mean of the numbers" << std::endl;
        std::cout << "S - Display the smallest" << std::endl;
        std::cout << "L - Display the largest" << std::endl;
        std::cout << "Q - Quit" << std::endl;
        std::cin >> selection;

        if (selection == 'p' || selection == 'P') {
            if (numbersList.size() == 0) {
                clearScreen();
                std::cout << "There are no numbers in the list. " << std::endl;
                std::cout << "==================================" << std::endl;
            } else {
```

```
                clearScreen();
                for (auto item : numbersList) {
                    std::cout << item << " ";
                }
                std::cout << std::endl;
                std::cout << "==================================" << std::endl;
            }
        } else if (selection == 'a' || selection == 'A') {
            int numberBuffer{};
            std::cout << "Enter the number to add to the vector: ";
            std::cin >> numberBuffer; // Circle back to this to deal with input
            validation
            numbersList.push_back(numberBuffer);
            clearScreen();
        } else if (selection == 'm' || selection == 'M') {
            double average{};
            for (auto item : numbersList) {
                average += item;
            }
            average /= numbersList.size();
            clearScreen();
            std::cout << "The average is: " << average << std::endl;
            std::cout << "================" << std::endl;

        } else if (selection == 's' || selection == 'S') {
            double swap{numbersList[0]};
            for (auto item : numbersList) {
                if (swap > item) {
                    swap = item;
                }
            }
            clearScreen();
            std::cout << "The smallest number is: " << swap << std::endl;
            std::cout << "================" << std::endl;
        } else if (selection == 'l' || selection == 'L') {
            double swap{ numbersList[0] };
            for (auto item : numbersList) {
                if (swap < item) {
                    swap = item;
                }
            }
            clearScreen();
            std::cout << "The largest number is: " << swap << std::endl;
            std::cout << "================" << std::endl;
        } else if (selection == 'q' || selection == 'Q') {
            clearScreen();
            std::cout << "Thanks for using the program." << std::endl;
        } else {
            clearScreen();
            std::cout << "That is not a valid selection, please try again." << std::endl;
        }
    } while (selection != 'q' && selection != 'Q'); // this kind of after - check needs
    the && condition
    return 0;
}
```