

```

1 //=====
2 // Section 9 Controlling Program Flow
3 //=====
4
5 // Controlling program flow
6 Sequence
7 - Ordering statements sequentially
8
9 Selection
10 - Making decisions
11
12 Iteration
13 - Looping or repeating
14
15 // Main statements
16 if statement
17 if-else statement
18 Nested if statements
19 switch statement
20 conditional operator ?: (ternary)
21
22 // Iteration
23 for loop
24 range-based
25 for loop
26 while loop
27 do-while loop
28 continue and break
29 infinite loop
30 nested loop
31
32 // The if statement
33 if (expression){Do something};
34 - if the expression is a boolean true, do the things
35 - if the expression is false, then skip the statement
36
37 If the statement to execute is a single line, you don't need the braces
38 The braces make it a 'block' statement
39
40 // The if/else statement
41 if (expression){
42     // If expression is true do this
43 } else {
44     // else if the expression is false do this
45 }
46
47 // The if/else if statement
48 if(expression){
49     // if true do this
50 } else if (expression){
51     // else if this is true do this
52 } else {
53     // Default
54 }
55
56 // Nested if statements
57 if(expression){
58     if (expression){
59     }
60 }
61
62
63
64
65
66
67
68
69

```

```

70 // Dangling else problem - which if does an else belong to?
71 if (expression)
72     if (expression)
73         // do something
74     else
75         // do something else
76
77 // In C++ the else belongs to the closest if statement
78 // use block statements to avoid this problem entirely
79
80 // Nicely formatting dollars
81 #include <iomanip>
82 setprecision(2); // Number of decimal points it will print
83
84 // Switch\case statement
85 // must evaluate to an integral or enumeration type - constants or literals known as
86 // compile time
87 // Numbers will work, character literals will also work
88
89 switch (integerControlExpression){
90     case expression1:
91         //statements
92         break;
93
94     case expression2:
95         //statements
96         break;
97
98     default:
99 }
100
101 If you omit the break statement the cases will fall through
102
103 // With an enumeration type
104 enum Color{
105     red,green,blue
106 };
107 Color screenColor{};
108
109 switch(screenColor){
110     case red:
111         //whatever
112     case green:
113         {
114             const int whatever{0};
115             break;
116         }
117
118     case blue:
119         //whatever
120         break;
121     default:
122         //whatever
123 }
124
125 // Switch statement facts
126 - Control expression must evaluate to an integer type
127 - Case expressions must be constant expressions that evaluate to integers or integer
128   literals
129 - Once a match occurs all following case statements are executed until a break is
130   encountered
131 - Provide a break statement for each case
132 - default is optional but should be handled
133 - note that if you need a variable in your case statements you need to create a block {}
134
135

```

```

136 // Conditional operator
137 (conditional expression) ? expression1 : expression2
138
139 The conditional expression must be a boolean value
140 If it's true, the first expression is returned.
141 If it's false, the second is returned
142 Similar to if\else
143 Ternary operator
144 Useful when used inline
145 very easy to abuse!
146
147 result = (a > b) ? a : b;
148
149 // Looping
150 - Third basic block of programming
151 - sequence, selection, iteration
152 Iteration or repetition
153 Allows the execution of a statement or block of statements repeatedly
154 Loops are made up of a loop condition and the body which contains the statements to
    repeat
155
156 // Use cases
157 Execute a loop:
158 - specific number of times
159 - for each element in a loop
160 - while a condition is true
161 - until a condition becomes false
162 - until we reach the end of some input stream
163 - forever
164 - many, many more
165
166 // Looping structures
167 for loop
168 - Iterate a specific number of times
169
170 range-based for loop
171 - one iteration for each element in a range or collection
172
173 while loop
174 - while a condition is true
175 - stops when it becomes false
176 - condition checked at the beginning of each iteration
177
178 do-while loop
179 - iterate while a condition is true
180 - stop when the condition becomes false
181 - checks at the end of every iteration
182
183 // The for loop
184 for (initialization statement; condition to test; increment){
185     //statements
186 }
187
188 (can also be written without the braces but only a single line)
189
190 for (int i{0}; i <= 10, i++){
191     std::cout << someArray[i] << endl;
192 }
193 NOTE: When we initialize the i inside the loop, it's only available inside the loop
194 NOTE: be careful of bounds checking - if you start at
195
196 // Compound loops
197 for(int i{}, int j{}; i < 10, j < 10; i++, j++){
198     // This is a really ugly syntax
199 }
200
201 NOTE: You may get a warning if you don't specify unsigned for the iterator and
202 you use something like the size() method on a vector which does use unsigned by default
    (no negative numbers);

```

```

203 // Range-based for loop
204 for (dataType variableName: sequence){
205     //statements
206 }
207
208 for (int items: itemArray){ // Use the same data type as the array
209     std::cout << items << std::endl;
210 }
211
212 for (auto items: itemArray){ // compiler will automatically decide
213     std::cout << items << std::endl;
214 }
215
216 // It's possible to use it with an inline collection
217 for(auto temp: {65,67,78,98}){
218 }
219
220 // You can also use it to iterate over a string
221 for (auto letters:"Josie"){
222
223 }
224
225 // More on <iomanip>
226 #include <iomanip>
227 cout << fixed << setprecision(1); // will set to 1 decimal
228
229 // While loop - this loop might never execute depending on the test condition
230 while (expression){
231     //statements
232 }
233
234 - You can create an infinite loop by doing:
235 while(true){
236     // This loop will never end unless you break out
237 }
238
239 - commonly used with boolean flags
240
241 // do while loop
242 do {
243     statements;
244 } while (expression)
245
246 This will execute at least once since we check at the end
247
248 // Continue and break
249 continue
250 - no further statements in the body of the loop are executed
251 - control immediately goes back to the beginning of the loop for the next iteration
252
253 break
254 - no further statements in the body of the loop are executed
255 - loop is terminated
256 - control immediately goes to the statement following the loop
257
258 // Infinite loops
259 Loops whose condition always evaluates to true
260 Usually it's a mistake
261 Sometimes programmers use them and include break statements to control them
262 Sometimes they are exactly what we need
263 - Event loop in an event-driven program
264 - Operating system
265
266
267 // Infinite for loop
268 for(;;){}
269
270 // Infinite while
271 while(true){}

```

```

272 // Infinite do while
273 do{}while(true);
274
275 // Example
276 while(true){
277     char again{};
278     cout << "Do you want to loop again?";
279     cin >> again;
280     if(again == "N" || again == 'n'){
281         break;
282     }
283 }
284
285 // Nested loops
286 Loop nested within another loop
287 Can be as many levels deep as the program needs
288 Very useful with multi-dimensional data structures
289 Outer loops vs inner loops - one iteration for outer then all the inner loops
290
291 #include <iostream>
292 int main(){
293     std::cout << "Multiplication tables below:" << std::endl;
294     std::cout << "=====" << std::endl;
295     for (int i{ 1 }; i <= 10; i++) {
296         for (int j{ 1 }; j <= 10; j++) {
297             std::cout << i << " * " << j << " = " << i * j << std::endl;
298         }
299         std::cout << "=====" << std::endl;
300     }
301 }
302
303 // Example iteration over a grid
304 int grid[5][3]{};
305
306 for (int row{0}; row < 6; row++){
307     for (int col{0}; col < 3; col++){
308         grid[row][col] = 1000;
309     }
310 }
311
312 //Example with vectors
313 vector<vector<int>> twoDimensionalVector{
314     {1,2,3},
315     {10,20,30,40},
316     {100,200,300,400,500}
317 };
318
319 for (auto row:twoDimensionalVector){
320     for(auto col:twoDimensionalVector){
321         std::cout << col << " ";
322     }
323     std::cout << endl;
324 }
325
326 //Example with vectors - this way we can use different sizes for the rows
327 std::vector<std::vector<int>> twoDimensionalVector{
328     {1,2,3},
329     {10,20,30,40},
330     {100,200,300,400,500}
331 };
332
333 for (auto row : twoDimensionalVector) {
334     for (auto col : row) { // In this loop we need to iterate over the previous loop
335         std::cout << col << " ";
336     }
337     std::cout << std::endl;
338 }
339
340

```

```

341 // Histogram example
342 #include <iostream>
343 #include <vector>
344 #include <string>
345
346 int main(){
347
348     int numberOfItems{};
349     std::cout << "How many data items do you have? ";
350     std::cin >> numberOfItems;
351
352     std::vector<int> data{};
353     for (int i{ 1 }; i <= numberOfItems; i++) {
354         int dataItem{};
355         std::cout << "Please enter the data to push into the vector: ";
356         std::cin >> dataItem;
357         data.push_back(dataItem);
358     }
359
360     std::cout << "Displaying histogram:" << std::endl;
361
362     for (auto item : data) {
363         for (int i{ 0 }; i < item; i++) {
364             if (i % 5 == 0) { // print a * for every 5 items
365                 std::cout << "*";
366             } else {
367                 std::cout << "-";
368             }
369         }
370         std::cout << std::endl;
371     }
372     std::cout << std::endl;
373     return 0;
374 }
375
376 //=====
377 // Section 9 Controlling Program Flow Challenge
378 //=====
379
380 #include <iostream>
381 #include <vector>
382 #include <string>
383
384 void clearScreen() {
385     // This translates to a code that clears the console
386     std::cout << "\033[2J\033[1;1H";
387 }
388
389 int main(){
390
391     char selection{};
392     std::vector<double> numbersList{};
393     std::string message{};
394     do {
395         std::cout << "Please make a selection: " << std::endl;
396         std::cout << "P - Print numbers" << std::endl;
397         std::cout << "A - Add a number" << std::endl;
398         std::cout << "M - Display the mean of the numbers" << std::endl;
399         std::cout << "S - Display the smallest" << std::endl;
400         std::cout << "L - Display the largest" << std::endl;
401         std::cout << "Q - Quit" << std::endl;
402         std::cin >> selection;
403
404         if (selection == 'p' || selection == 'P') {
405             if (numbersList.size() == 0) {
406                 clearScreen();
407                 std::cout << "There are no numbers in the list. " << std::endl;
408                 std::cout << "===== " << std::endl;
409             } else {

```

```

410         clearScreen();
411         for (auto item : numbersList) {
412             std::cout << item << " ";
413         }
414         std::cout << std::endl;
415         std::cout << "======" << std::endl;
416     }
417 } else if (selection == 'a' || selection == 'A') {
418     int numberBuffer{};
419     std::cout << "Enter the number to add to the vector: ";
420     std::cin >> numberBuffer; // Circle back to this to deal with input
                                validation
421     numbersList.push_back(numberBuffer);
422     clearScreen();
423 } else if (selection == 'm' || selection == 'M') {
424     double average{};
425     for (auto item : numbersList) {
426         average += item;
427     }
428     average /= numbersList.size();
429     clearScreen();
430     std::cout << "The average is: " << average << std::endl;
431     std::cout << "======" << std::endl;
432
433 } else if (selection == 's' || selection == 'S') {
434     double swap{numbersList[0]};
435     for (auto item : numbersList) {
436         if (swap > item) {
437             swap = item;
438         }
439     }
440     clearScreen();
441     std::cout << "The smallest number is: " << swap << std::endl;
442     std::cout << "======" << std::endl;
443 } else if (selection == 'l' || selection == 'L') {
444     double swap{ numbersList[0] };
445     for (auto item : numbersList) {
446         if (swap < item) {
447             swap = item;
448         }
449     }
450     clearScreen();
451     std::cout << "The largest number is: " << swap << std::endl;
452     std::cout << "======" << std::endl;
453 } else if (selection == 'q' || selection == 'Q') {
454     clearScreen();
455     std::cout << "Thanks for using the program." << std::endl;
456 } else {
457     clearScreen();
458     std::cout << "That is not a valid selection, please try again." << std::endl;
459 }
460 } while (selection != 'q' && selection != 'Q'); // this kind of after - check needs
the && condition
461 return 0;
462 }

```