

CS331
Program 1
James Francis
User 12

```

cat -b lab1.h
1 #ifndef LAB1_H
2 #define LAB1_H

3 #include <iostream>
4 #include <fstream>
5 #include <math.h>
6 using namespace std;

7 class lab1{

8 public:
9     lab1();
10 bool checkValueTwo(int, int, int);
11 bool checkValueThree(int, int, int, int);
12 bool checkValueFour(int, int, int, int, int);
13 string swapNumforDarkChar(int);
14 };
15 #endif
:
printf "\\n\\n"

```

```

cat -b lab1.cpp
1 /*
2 PROGRAM NAME: Program 1: Developing skills for 2d arrays in C++
3
4 PROGRAMMER:   James Francis
5
6 CLASS:        CSC 331.001, Fall 2014
7
8 INSTRUCTOR:   Dr. Strader.
9
10 DATE STARTED: September 9, 2014
11
12 DUE DATE:     September 11, 2014
13
14 PROGRAM PURPOSE:
15
16 This program will read in values from a .dat file, store those values in a 2D array. It will then output
17 that 2D array and a Dark Character representation of the input to the console.
18 Then the program will check the nearest neighbor values (+/-)1 both laterally and vertically, if the
19 current element differs from these values by more than one, the program will take the average of the
20 neighbor values, rounded to the nearest integer and assign that value to the current element.
21
22 Once this "error checking" has been completed, the Corrected Input will be printed to console along with

```

```

23 a Dark Character representation of the Corrected Input.
24
25
26
27 VARIABLE DICTIONARY:
28
29 int size: size of the 2 dimensional array, determined by the first line of input file
30 int left, right, top, bottom: values of the elements surrounding the current element
31 int intArray[][]: 2 dimensional array used to hold the input
32 string dashes, spaces: strings used solely for formatting, to make output look presentable(?)

33
34 ADTs: none
35
36 FILES USED:
37
38 prog1.dat
39
40
41
42 SAMPLE INPUTS:
43
44 10
45 7 6 9 4 5 4 3 2 1 0
46 6 5 5 5 6 5 4 3 2 1
47 6 5 6 6 7 6 8 4 3 2
48 1 5 6 7 7 7 6 5 4 3
49 5 5 6 7 6 7 7 6 5 9
50 5 6 7 6 5 6 6 5 4 3
51 5 6 7 9 5 5 6 5 4 3
52 5 5 6 7 6 6 7 6 5 4
53 5 9 5 6 7 6 5 0 3 2
54 5 5 5 5 6 5 4 3 2 7

55
56
57 SAMPLE OUTPUTS:
58
59 Uncorrected Input
60 -----|
61 7 6 9 4 5 4 3 2 1 0 |
62 6 5 5 5 6 5 4 3 2 1 |
63 6 5 6 6 7 6 8 4 3 2 |
64 1 5 6 7 7 7 6 5 4 3 |
65 5 5 6 7 6 7 7 6 5 9 |
66 5 6 7 6 5 6 6 5 4 3 |

```

```

67 5 6 7 9 5 5 6 5 4 3 |
68 5 5 6 7 6 6 7 6 5 4 |
69 5 9 5 6 7 6 5 0 3 2 |
70 5 5 5 5 6 5 4 3 2 7 |
71 |
72 # * & ! + ! - , . |
73 * + + + * + ! - , . |
74 * + * * # * $ ! - , |
75 . + * # # # * + ! - |
76 + + * # * # # * + & |
77 + * # * + * * + ! - |
78 + * # & + + * + ! - |
79 + + * # * * # * + ! |
80 + & + * # * + - , |
81 + + + + * + ! - , # |
82 -----|
83
84
85 Corrected Input
86 -----|
87 7 7 5 4 5 4 3 2 1 0 |
88 6 6 5 5 6 5 5 3 2 1 |
89 4 5 6 6 7 7 6 4 3 2 |
90 5 5 6 7 7 7 6 5 4 5 |
91 5 5 6 7 6 7 7 6 6 5 |
92 5 6 7 7 6 6 6 5 5 4 |
93 5 6 7 7 6 5 6 5 4 3 |
94 5 7 6 7 6 6 6 4 4 3 |
95 6 6 5 6 7 6 4 4 3 4 |
96 5 5 5 5 6 5 4 3 4 4 |
97 |
98 # # + ! + ! - , . |
99 * * + + * + + - , . |
100 ! + * * # # * ! - , |
101 + + * # # # * + ! + |
102 + + * # * # # * * + |
103 + * # # * * * + + ! |
104 + * # # * + * + ! - |
105 + # * # * * * ! ! - |
106 * * + * # * ! ! - ! |
107 + + + + * + ! - ! ! |
108 -----|

109 -----*/
110 #include "lab1.h"

```

```

111 lab1::lab1(){
112     int size = 0;
113     int left=0;
114     int right=0;
115     int bottom=0;
116     int top = 0;
117
118     //-----
119     // Open input file and determine the size of the the array by assigning the first valeu in the file to
120     // the size variable
121     //-----
122
123     ifstream infile("../instr/prog1.dat");
124     for (int i=0;i<1;i++){
125         infile>>size;
126         cout<<endl;
127     }
128     int intArray[size][size];
129     //-----
130     // Populate the previously created intArray[][] of dimesions [size] x [size]
131     //-----
132     for (int i = 0; i<size; i++){
133         for(int j = 0; j<size; j++){
134             infile>>intArray[i][j];
135         }
136     }
137
138     string spaces = "";
139     string dashes = "-";
140
141     //-----
142     // Begin segment of code to print uncorrected input and corresponding dark character for that input
143     //-----
144
145     cout << "Uncorrected Input" <<std::endl;
146
147     for (int i = 0; i <size;i++){
148         dashes=dashes+"--";
149         spaces=spaces+" ";
150     }
151
152     cout << dashes << "| " <<std::endl;
153     for (int i = 0; i < size; ++i){
154         for (int j = 0; j < size; ++j){

```

```

156         std::cout << intArray[i][j] << ' ';
157     }
158     std::cout << " | " <<std::endl;
159 }
160
161 cout << spaces <<" | " <<std::endl;
162
163 for (int i = 0; i < size; ++i){
164     for (int j = 0; j < size; ++j){
165         int value = intArray[i][j];
166         std::cout << lab1::swapNumforDarkChar(value) << ' ';
167     }
168     std::cout << " | " <<std::endl;
169 }
170
171 cout << dashes << "| " <<std::endl;
172 cout <<std::endl<<std::endl;
173
174
175
176 //-----
177 // Begin segment of code to print Corrected Input and corresponding dark characters
178 //-----
179
180 cout << "Corrected Input" << std::endl;
181
182 cout << dashes << "| " <<std::endl;
183 int current = 0;
184 for (int i = 0; i < size; i++){
185     for (int j = 0; j < size; j++){
186         current = 0;
187         double avg=0;
188         current = intArray[i][j];
189
190         //-----
191         //This statement handles the special case of the first element within the first row
192         //-----
193         if (i==0 && j==0) {
194             //-----
195             //This statement handles the special case of the first element within the first row
196             //-----
197             right = intArray[i][j+1];
198             bottom = intArray[i+1][j];
199
200             int difference1 = std::abs(current-bottom);
201             int difference2 = std::abs(current-right);

```

```

202
203         if((difference1==1 && difference2==1) || (difference1==0&&difference2 ==0) || (difference1==1 &&
difference2==0)|| (difference1==0 && difference2==1)){
204             intArray[i][j] = intArray[i][j];
205
206         }
207         else{
208             //-----
209             // This statement is run if any of the difference values are more than 1
210             //-----
211             avg = (right+bottom)/2.0;
212             intArray[i][j] = static_cast<int>(floor(avg+.5));
213         }
214     }
215
216
217     //-----
218     //This statement handles the special case of the nonfirst and nonlast elements in the
219     //first row
220     //-----
221     if (i==0 && (j>0 && j<(size-1))) {
222         //-----
223         //This statement handles the special case of the nonfirst and nonlast elements in the
224         //first row
225         //-----
226
227         left = intArray[i][j-1];
228
229
230         right = intArray[i][j+1];
231
232         bottom = intArray[i+1][j];
233
234         if(lab1::checkValueThree(current,left, right, bottom)){
235             intArray[i][j] = intArray[i][j];
236
237         }
238         else{
239             //-----
240             // This statement is run if checkValueThree is false
241             //-----
242             avg =(left+right+bottom)/3.0;
243             intArray[i][j] = static_cast<int>(floor(avg+.5));
244
245         }
246     }

```

```

247
248 //-----
249 //This statement handles the special case of the last element within the first row
250 //-----
251 if (i==0 && j==size-1) {
252     //-----
253     //This statement handles the special case of the last element within the first row
254     //-----
255
256     left = intArray[i][j-1];
257     bottom = intArray[i+1][j];
258     if(lab1::checkValueTwo(current, left, bottom)==true){
259         intArray[i][j] = intArray[i][j];
260
261     }
262     else{
263         intArray[i][j] = static_cast<int>(floor(((left+bottom)/2.0)+.5));
264     }
265 }
266
267 //-----
268 //This statement handles the special case of nonfirst and nonlast elements in the
269 //first column
270 //-----
271 if ( (i>0 && i<(size-1)) && j==0) {
272     //-----
273     //This statement handles the special case of nonfirst and nonlast elements in the
274     //first column
275 //-----
276
277     top = intArray[i-1][j];
278     right = intArray[i][j+1];
279
280     bottom = intArray[i+1][j];
281
282     if(lab1::checkValueThree(current, top, right, bottom)==true){
283         intArray[i][j] = intArray[i][j];
284
285     }
286     else{
287         intArray[i][j] = static_cast<int>(floor(((top+right+bottom)/3.0)+.5));
288
289     }
290 }
291
292 //-----

```



```

293 //This statement handles all elements not in the first or last row or column
294 //-----
295 if ( (i>0 && i<(size-1)) && (j>0&&j<(size-1)) ){
296 //-----
297 //This statement handles all elements not in the first or last row or column
298 //-----
299
300     top = intArray[i-1][j];
301     left = intArray[i][j-1];
302     right = intArray[i][j+1];
303     bottom = intArray[i+1][j];
304     if(lab1::checkValueFour(current, top, left, right, bottom)){
305         intArray[i][j] = intArray[i][j];
306     }
307     else{
308         intArray[i][j] = static_cast<int>(floor(((top+left+right+bottom)/4.0)+.5));
309     }
310 }
311
312 //-----
313 //This statement handles the special case of last elements in the last column
314 //-----
315 if ( (i>0 && i<(size-1)) && j==(size-1)) {
316 //-----
317 //This statement handles the special case of last elements in the last column
318 //-----
319
320     top = intArray[i-1][j];
321     left = intArray[i][j-1];
322     bottom = intArray[i+1][j];
323     if(lab1::checkValueThree(current,top, left, bottom)){
324         intArray[i][j] = intArray[i][j];
325     }
326     else{
327         intArray[i][j] = static_cast<int>(floor(((top+left+bottom)/3.0)+.5));
328     }
329 }
330
331 //-----
332 //This statement handles the special case of the first element within the last row
333 //-----
334 if (i==(size-1) && j==0) {
335 //-----
336 //This statement handles the special case of the first element within the last row
337 //-----
338     top = intArray[i-1][j];

```

```

339         right = intArray[i][j+1];
340         if(lab1::checkValueTwo(current, right, top)){
341             intArray[i][j] = intArray[i][j];
342         }
343         else{
344             intArray[i][j] = static_cast<int>(floor(((right+top)/2.0)+.5));
345         }
346     }
347
348     //-----
349     //This statement handles the special case of the nonfirst and nonlast elements in the
350     //last row
351     //-----
352     if (i==(size-1) && (j>0&&j<(size-1))) {
353         //-----
354         //This statement handles the special case of the nonfirst and nonlast elements in the
355         //last row
356         //-----
357
358         left = intArray[i][j-1];
359         right = intArray[i][j+1];
360         top = intArray[i-1][j];
361         if(lab1::checkValueThree(current,left, right, top)){
362             intArray[i][j] = intArray[i][j];
363         }
364         else{
365             intArray[i][j] = static_cast<int>(floor(((left+right+top)/3.0)+.5));
366         }
367     }
368
369     //-----
370     //This statement handles the special case of the first element within the first row
371     //-----
372     if (i==size-1&&j==size-1) {
373         //-----
374         //This statement handles the special case of the first element within the first row
375         //-----
376         top = intArray[i-1][j];
377         left = intArray[i][j-1];
378         if(lab1::checkValueTwo(current, top, left)){
379             intArray[i][j] = intArray[i][j];
380
381         }
382         else{
383             intArray[i][j] = static_cast<int>(floor(((top+left)/2.0)+0.5));
384         }

```

```

385         }
386         std::cout<< intArray[i][j]<< ' ';
387
388     }
389 }
390
391     std::cout << " | " <<std::endl;
392 }
393
394     std::cout << spaces <<" | " <<std::endl;
395
396
397     //-----
398     // The following loop outputs the dark characters for the corrected intArray
399     //-----
400     for (int i = 0; i < size; ++i){
401         for (int j = 0; j < size; ++j){
402             int value = intArray[i][j];
403             std::cout << lab1::swapNumforDarkChar(value) << ' ';
404         }
405         std::cout << " | " <<std::endl;
406     }
407     std::cout << dashes << "| " <<std::endl;
408     std::cout <<std::endl<<std::endl;
409
410 }
411 bool lab1::checkValueTwo(int current, int val1, int val2){
412     //-----
413     //This method compares a case in which there are only two bordering values to a current value
414     //if the difference between the current value and the other two values is greater than one,
415     //false is returned to the calling code.
416     //
417     //-----
418
419     bool result = true;
420
421     int difference1 = std::abs(current-val1);
422     int difference2 = std::abs(current-val2);
423
424     if (difference1 > 1||difference2 > 1) {
425         result = false;
426         return result;
427     }else return result;
428 }

```

```

429 bool lab1::checkValueThree(int current, int val1, int val2, int val3){
430     //-----
431     //This method compares a case in which there are three bordering values to a current value.
432     //If the difference between the current value and any of the other three values is greater
433     //than one, false is returned to the calling code.
434     //-----
435
436     bool result = true;
437
438     int difference1 = abs(current-val1);
439     int difference2 = abs(current-val2);
440     int difference3 = abs(current-val3);
441
442     if (difference1 > 1){
443         return result = false;
444     }
445     else if (difference2 > 1) {
446         return result = false;
447     } else if (difference3 > 1) {
448         return result = false;
449     }else
450
451         return result;
452 }

453 bool lab1::checkValueFour(int current, int val1, int val2, int val3, int val4){
454     //-----
455     //This method compares a case in which there are four bordering values to a current value.
456     //If the difference between the current value and any of the other four values is greater
457     //than one, false is returned to the calling code.
458     //
459     //-----
460
461     bool result = true;
462
463     int difference1 = std::abs(current-val1);
464     int difference2 = std::abs(current-val2);
465     int difference3 = std::abs(current-val3);
466     int difference4 = std::abs(current-val4);
467
468     if (difference1>1||difference2>1||difference3>1||difference4>1) {
469
470         return result = false;
471     }else
472
473         return result;

```

```
474 }
```

```
475 string lab1::swapNumforDarkChar(int array){
```

```
476     //-----  
477     //This method compares the integer input from the calling code, and returns a string.  
478     //The string returned is the corresponding "Dark Character" representation of the input value.  
479     //-----
```

```
480     string result = " ";
```

```
481     if(array== 0){
```

```
482         result = " ";
```

```
483     }
```

```
484     else if(array== 1){
```

```
485         result = ".";
```

```
486     }
```

```
487     else if(array== 2){
```

```
488         result = ",";
```

```
489     }
```

```
490     else if(array== 3){
```

```
491         result = "-";
```

```
492     }
```

```
493     else if(array== 4){
```

```
494         result = "!";
```

```
495     }
```

```
496     else if(array== 5){
```

```
497         result = "+";
```

```
498     }
```

```
499     else if(array== 6){
```

```
500         result = "*";
```

```
501     }
```

```
502     else if(array== 7){
```

```
503         result = "#";
```

```
504     }
```

```
505     else if(array== 8){
```

```
506         result = "$";
```

```
507     }
```

```
508     else if(array== 9){
```

```
509         result = "&";
```

```
510     }
```

```
511     return result;
```

```
512 }
```

```
:  
printf "\\n\\n
```

```

cat -b lab1test.cpp
 1 //
 2 //  lab1test.cpp
 3 //  prog1
 4 //
 5 //  Created by James Francis II on 9/9/14.
 6 //  Copyright (c) 2014 James Francis II. All rights reserved.
 7 //

 8 #include "lab1.h"

 9 int main(int argc, const char * argv[])
10 {
11     lab1();
12     return 0;
13 }
:
g++ lab1.cpp lab1test.cpp -o p1
:
p1

```

Uncorrected Input

```

-----|
7 6 9 4 5 4 3 2 1 0 |
6 5 5 5 6 5 4 3 2 1 |
6 5 6 6 7 6 8 4 3 2 |
1 5 6 7 7 7 6 5 4 3 |
5 5 6 7 6 7 7 6 5 9 |
5 6 7 6 5 6 6 5 4 3 |
5 6 7 9 5 5 6 5 4 3 |
5 5 6 7 6 6 7 6 5 4 |
5 9 5 6 7 6 5 0 3 2 |
5 5 5 5 6 5 4 3 2 7 |
|
# * & ! + ! - , . |
* + + + * + ! - , . |
* + * * # * $ ! - , |
. + * # # # * + ! - |
+ + * # * # # * + & |
+ * # * + * * + ! - |
+ * # & + + * + ! - |
+ + * # * * # * + ! |
+ & + * # * + - , |
+ + + + * + ! - , # |
-----|

```

Corrected Input

```
-----|
7 7 5 4 5 4 3 2 1 0 |
6 6 5 5 6 5 5 3 2 1 |
4 5 6 6 7 7 6 4 3 2 |
5 5 6 7 7 7 6 5 4 5 |
5 5 6 7 6 7 7 6 6 5 |
5 6 7 7 6 6 6 5 5 4 |
5 6 7 7 6 5 6 5 4 3 |
5 7 6 7 6 6 6 4 4 3 |
6 6 5 6 7 6 4 4 3 4 |
5 5 5 5 6 5 4 3 4 4 |
-----|

# # + ! + ! - , . |
* * + + * + + - , . |
! + * * # # * ! - , |
+ + * # # # * + ! + |
+ + * # * # # * * + |
+ * # # * * * + + ! |
+ * # # * + * + ! - |
+ # * # * * * ! ! - |
* * + * # * ! ! - ! |
+ + + + * + ! - ! ! |
-----|
```