

The screenshot shows a C++ IDE with a source editor and a console window. The source editor contains the following code:

```

49
50 void markSrt(int a[],int n){
51     for(int pos=0;pos<n-1;pos++){
52         for(int i=pos+1;i<n;i++){
53             if(a[pos]>a[i]){
54                 int temp=a[i];
55                 a[i]=a[pos];
56                 a[pos]=temp;
57             }
58         }
59     }
60 }
61
62 void prntAry(int array[],int n,int perLine){
63     cout<<endl;
64     for(int i=0;i<n;i++){
65         cout<<array[i]<<" "; //2 digit random number
66         if(i%perLine==(perLine-1))cout<<endl;
67     }
68     cout<<endl;
69 }
70
71 void fillAry(int array[],int n){
72     for(int i=0;i<n;i++){
73         array[i]=rand()%90+10; //2 digit random number
74     }
75 }

```

The console window shows the output of the program, displaying two rows of 10 numbers each, followed by a 10x10 grid of numbers. The output is as follows:

```

82 83 14 89 32 77 35 62 29 49
93 23 22 60 79 58 84 30 19 73

12 13 13 13 13 14 14 14 14 15
17 19 19 20 21 21 22 22 23 23
24 24 24 25 25 26 27 29 29 30
31 32 32 33 35 35 36 36 37 38
38 39 40 42 45 46 46 47 48 49
49 51 52 53 56 58 59 59 60 61
61 62 62 62 62 65 66 67 67 71
71 72 73 73 77 77 78 79 81 82
82 83 83 84 84 84 84 84 85 85
89 89 89 93 93 94 95 95 97 97

RUN FINISHED; exit value 0; real time: 10ms; user: 0ms; system: 0ms

```

Figure 1. Using a single array.

The screenshot shows a C++ IDE with three tabs: `main.cpp`, `main.cpp`, and `Array.h`. The code in the editor is as follows:

```
72
73 void prntAry(int array[],int indx[],int n,int perLine){
74     cout<<endl;
75     for(int i=0;i<n;i++){
76         cout<<array[indx[i]]<<" "; //2 digit random number
77         if(i%perLine==(perLine-1))cout<<endl;
78     }
79     cout<<endl;
80 }
81
82 void prntAry(int array[],int n,int perLine){
83     cout<<endl;
84     for(int i=0;i<n;i++){
85         cout<<array[i]<<" "; //2 digit random number
86         if(i%perLine==(perLine-1))cout<<endl;
87     }
88     cout<<endl;
89 }
90
91 void fillAry(int array[],int indx[],int n){
92     for(int i=0;i<n;i++){
93         array[i]=rand()%90+10; //2 digit random number
94         indx[i]=i;
95     }
96 }
```

The output window shows the following text:

```
95 77 42 53 45 55 87 58 15 14
99 72 68 37 98 93 20 18 47 58
93 64 27 22 20 29 10 41 15 16

The indexed Array before sorting.

0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99

The original Array using index before sorting.
```

Figure 2. Using another array as an index.

```
63 }
64 }
65
66 void prntAry(Array *array,int perLine){
67     cout<<endl;
68     for(int i=0;i<array->size;i++){
69         cout<<array->data[i]<<" "; //2 digit random number
70         if(i%perLine==(perLine-1))cout<<endl;
71     }
72     cout<<endl;
73 }
74
75 Array *fillAry(int n){
76     //Allocate memory
77     Array *array=new Array;
78     array->size=n;
79     array->data=new int[n];
80     //Fill with random values
81     for(int i=0;i<n;i++){
82         array->data[i]=rand()%90+10; //2 digit random number
83     }
84     //return the array
85     return array;
86 }
```

fillAry

ort (Build. Run) ×	MarkSort (Run) ×	MarkSortUsinalIndex (Build. Run) ×	MarkSortUsinalIndex (Run) ×	MarkSortP					
58	85	96	23	71	56	40	79	55	80
66	91	56	70	26	30	23	11	89	93
81	13	17	86	95	82	86	83	29	37
54	77	23	50	52	46	58	45	77	65
25	43	18	72	65	34	92	79	87	43
10	10	11	13	14	17	18	18	18	19
21	23	23	23	23	23	25	26	26	28
29	30	30	33	34	34	35	35	36	37
37	40	40	43	43	45	45	46	50	52
52	54	55	56	56	58	58	60	65	65
65	65	65	66	69	70	71	72	73	75
76	77	77	79	79	80	80	81	81	82
83	84	85	85	85	86	86	86	87	87
88	88	88	89	89	89	91	91	92	92
93	94	95	95	96	96	96	96	98	99

Figure 3. Using a structure and pointer to implement an array.

```
1  /*
2  * File:   main.cpp
3  * Author: Dr. Mark E. Lehr
4  * Created on May 26rd, 2019, 10:20 AM
5  * Purpose: Arrays with Class
6  */
7
8  //System Libraries
9  #include <iostream> //Input/Output Library
10 #include <cstdlib> //Random function
11 #include <ctime> //Time Library
12 using namespace std;
13
14 //User Libraries
15 #include "Array.h"
16
17 //Global Constants, no Global Variables are allowed
18 //Math/Physics/Conversions/Higher Dimensions - i.e. PI, e, etc...
19
20 //Function Prototypes
21 void prntAry(Array *,int);
22 void markSrt(Array *);
23
24 //Execution Begins Here!
25 int main(int argc, char** argv) {
26     //Set the random number seed
27     srand(static_cast<unsigned int>(time(0)));
28
29     prntAry(>
```

```
index (Build. Run) x MarkSortUsinIndex (Run) x MarkSortPointersStructure (Build. Run) x MarkSortPointersStructu
19 18 17 16 15 14 13 12 11 10
9 8 7 6 5 4 3 2 1 0

0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99

RUN FINISHED; exit value 0; real time: 0ms; user: 0ms; system: 0ms
```

Figure 4. Using a class to implement an array.

Lab052319 and Lab052619 illustrates a sorting algorithm using different implementations of an array data. Code 1 uses the basic single array datatype. Code 2 uses 2 single arrays where the second array is used as an index array. Only the index array is updated to sort the data in the first array. Code 3 uses a structure database and pointer to implement an array. The structure contains 2 data types, the size and array data. Size is implemented as an int while the array day is implemented as a basic single array of ints. Code 4 uses a class to implement an array. The class contains the data type and functions such as get Data and setData. Using classes illustrates object-oriented programming, wherein here the array is now an object with its own data and functions.