"The budget is not just a collection of numbers, but an expression of our values and aspirations."
- Jacob Lew

"I think that's what scares me: the randomness of everything. That the people who could be important to you might just pass you by. Or you pass them by." - Peter Cameron

Chapter 8

References, Randomization, Arrays and Filing

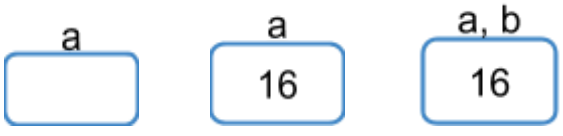
(Sorting, Binary Search, Statistical measures, and Electronic Voting)

Summary
This chapter takes you to build one of the most powerful constructs to store data called list or arrays. The sole objective is to make you equip with dealing with more real life problems where you actually needs to work generically on the list of data entries and access them for different types of measurements processing of real life data, which comes in abundance and storing it in some variables is impossible. This Array data structure will allow you flexibility of accessing the data in such a way that it combines all the learnt constructs in the most simplified manner with a lot of generalized version of implementations. This construct will become the base of solving so many amazing games like TicTactoe, Conway Game of life, Snakes and Ladders and Chess Game.

8.1 Aliasing/Reference Variable

- As we know that we can return only one value from a function what if we needed to return multiple values from a function?
- Until now we can not return multiple values from a function but from now on we will learn a new concept or technique of aliasing a variable. We all know that a variable is small place in a memory to store data it has a unique name, what if we name twice a single variable that is aliasing or referencing a variable. We know the syntax of initializing a variable like **int a,** if we want to make an alias of **a** just like this **int&b** this will make a second name for variable **a** that is **b**. From now own whatever the value of **a** is same value will be of **b** like we change value of **a** like **a=16;** and we output the value of **b** it will show 16 on console screen. Now if we change value of **b** what will happen? As it is the second name of **b** so value of **a** will also changed. Now, we come on our point that returning multiple values from a function actually there is no returning multiple value from a function we just alias the variable/s in function to on operate them or change value there it's actually like returning multiple value from a function.

Here the pictorial view that what will happen if we reference a variable

int a ; **a = 16;** **int & b = a;**	

Now the variable **a** has another name **b** you can access this location by calling it **a** or **b**.
One more thing you can make reference of any kind of variable. Actually its main application is to access a memory location outside its scope by making reference in other scope where we have to access.(We can also called it a portal to another world(scope)).
2. Its Applications: 2-Example of multiple value returning function Such as Divide for returning two values quotient and remainder…. Or any other example you want? That IFs chapter where 5 sections names and

Averages are passed and we needed to find which section got the highest Average/Aggregate and what Average/Aggregate?

Code

```
#include <iostream>
using namespace std;
int GCD(int n,int d);
int Smaller(int n1,int n2);
void ReduceFraction(int&n,int&d);
int main()
{
   int numerator,denominator;
   //this character is for saving the sign
   char ch;
   cout << "Enter a equation i.e 3/6: ";
   cin>>numerator
     >>ch
     >>denominator;
   ReduceFraction(numerator,denominator);
   cout<<"Your reduced fraction is "<<numerator<<"/"<<denominator<<endl;
   return 0;
}
int GCD(int n,int d)
{
   int result=0;
   int limit=Smaller(n,d);
   for(int cnt=1;cnt<=limit;cnt++)
   {
      if(n%cnt==0&&d%cnt==0)
      {
         result=cnt;
      }
   }
    return result;
}
void ReduceFraction(int&n,int&d)
{
   int gcd=0;
   gcd=GCD(n,d);
   n=n/gcd;
   d=d/gcd;
}
int Smaller(int n1,int n2)
{
   if(n1<n2)
      return n1;
   return n2;
}
```

Link: http://codepad.org/qx33APSS

Output

```
Enter a equation i.e 3/6: 11/99
Your reduced fraction is 1/9
```

Description

Remainder/Quotient
Code

```
#include <iostream>
using namespace std;
void divide(int&dvdnt,int&dvsr,int&rem,int&quo);
int main()
{
   int dividend,divisor,remainder,quotient;
   cout << "Enter the dividend and quotient :";
   cin>>dividend
     >>divisor;
   divide(dividend,divisor,remainder,quotient);
   cout<<"answer is "<<quotient<<" and remainder is "<<remainder<<endl;
   return 0;
}
void divide(int&dvdnt,int&dvsr,int&rem,int&quo)
{
   rem=dvdnt%dvsr;
   quo=dvdnt/dvsr;
}
```

Link http://codepad.org/SWsU5egu
Output

```
Enter the dividend and quotient :25 5
answer is 5 and remainder is 0
```

Description

8.2 Populating Randomization and experimentations

1. rand( ) - Function

rand( ) is a builtin function of **C++,** which returns a pseudo-random integral number in the range of 0 to **RAND_MAX** is a constant defined in **<cstdlib>. Header file** for **rand( )** function is **#include<cstdlib>** or **#include<time.h>**. According to standard library implementation, **RAND_MAX** is **32767**. So, it is fixed that the number which **rand( )** function will return\generate is less than 32767. Random number is generated by an algorithm that returns a sequence of apparently non-related numbers each time it is called.

Code

```
#include<iostream>
#include<cstdlib>
using namespace std;
int main()
{
        cout << rand( ) << endl;
        cout << rand( ) << endl;
        cout << rand( ) << endl;
        return 0;
}
```

Output of 1st run

```
41
18467
6334
```
1

Output of 2nd run

```
41
18467
6334
```

Output of 3rd run

```
41
18467
6334
```

Link:

Description

In above code, from line 1 to 3 we have used some c++ libraries. **#include<cstdlib>** is for **rand()** function. At line 4 **int main( )** starts. From line 6 to 8 we have displayed some randomly generated values at console which **rand( )** has returned, and at line 9 we used **return 0** which terminates our program. As we see, every time when we run the program it produce same output. We will discuss in a while why is this happening.

Here **RAND_MAX** whose value is library-dependent, but is guaranteed to be at least **32767** on any standard library implementation. If we want to set a range in between our demand, which we will get a randomly generated number then we will use **mod(%),** which is as followed:

```
1   #include<iostream>
2   #include<cstdlib>
3   using namespace std;
4   int main()
5   {
6           int v1 = rand( ) % 100;      // v1 in the range 0 to 99
7           int v2 = rand( ) % 100 + 1;          // v2 in the range 1 to 100
8           int v3 = rand( ) % 30 + 1985;  // v3 in the range 1985-2014
9           cout<<"v1= "<<v1<<endl;
10          cout<<"v2= "<<v2<<endl;
11          cout<<"v3= "<<v3<<endl;
12          return 0;
13  }
```

Output

```
v1= 41
v2= 68
v3= 1989
```

Description

In above code, from line 1 to 3 we used header files as we used before. At line 4 **int main()** starts. At line 6, we use **rand() % 100** it means we give the limit to the program that will only produce any number in the range of **0 to 99.** At line 7, this will generate number range of **1 to 100** and on next line it will produce number range of **1985-2014**. We can generate any range of number just by some calculations as like as we did in above code.

Algorithm of Rand( )

(using polynomials)
When RAND function is called then On backend, one polynomial is generated e.g. **(f(x)^2+g(x)+c)**. Here, “x” is seed. “Seed” is an initial value to start the process of generating the random numbers. After some calculations from polynomial, it generate a very large number. So, that`s why compiler Further takes mod of this large number that`s the way compiler may gives us a small random number that we want.

2. srand(int Seed) Function

As we know that, rand() generates same sequence of random numbers at every time when we run our code because of same seed stored in a program. So here, we can use **srand(int seed)** function by which we can change seed for generating random numbers using same header file as we used before for **rand()** function. **srand()** function takes one integer value called seed as a parameter and return one integer value which this function will generated randomly. But now here the difference is that we can change the seed at run time or at compile time.

Remember
The algorithm of generating random number uses seed to generate the number. But once seed is set in a program then it can't be change after compile time but this is user choice wherever he gives the seed to the function, at a compile time or at run time.

Code

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <time.h>>
4  using namespace std;
5  int main()
6  {
7     int seed=0;
8     cout << "Enter the seed : ";
9     cin >> seed;
10    srand( seed ) ;
11    int random_no = rand();
12
13    cout << "Random Number = " << random_no << endl ;
14    return 0;
15 }
```

Output of 1st run

```
Enter the seed : 5
Random Number = 54
```

Output of 2nd run

```
Enter the seed : 7
Random Number = 61
```

Output of 3rd run

```
Enter the seed : 2
Random Number = 45
```

Link:

Description
In above code, this program is just ask you the seed and generate random number on the base of seed which is enter by the user and shows us on console. It is clearly shown that on every seed there is different number is generated.

3. time(0) Function

For **srand(int)** function you have you provide seed for execution otherwise your code will not run, it will give you compile time error and also there is sequence behind every seed. So what if someone already knows the sequence behind the seed. For example, you write a code for game in which dice is used, then what if someone already knows that on which seed he wins then he will definitely choose that seed and will win everytime. So by avoiding that kind of unfair thing C++ allow us to generate truly a random number by **time(0)** function.
time(0) function generates the random number which is unpredictable but you can get whatever number you want, by taking mod as like as we did before in **rand()** function topic . Actually, **time(0)** function return us the time of system in seconds. When you mention **srand(time(0))** on the top of the int main then you don't need to enter seed every time when you run the code, it will calculate automatically “seed” for generating random number. Now you will not get the same sequence at every time of execution because time function will not give you the same time when you run your code again; time changes after every second so it is impossible that you will get the same time as seed again and again. If you think a little bit then you will know that **time(0)** function makes your life so easy for generating random numbers.

Remember
For use time(0) you must have to add a prototype #include<ctime> or #include<time.h> at start of code where other prototypes are used.

Example of **time(0)** is as followed:
Code

```
1  #include <iostream>
2  #include<cstdlib>
3  #include<time.h>
4  using namespace std;
5  int main()
6  {
7     srand( time(0) );
8     int random_no;
9     random_no = rand();
10
11    cout << "Random number = " << random_no << endl ;
12    return 0;
13 }
```

Output of 1st run

```
Random number = 28345
```

Output of 2nd run

```
Random number = 28606
```

Output of 3rd run

```
Random number = 28701
```

Link:

Description

In this program, at line 9, this statement generates randomly seed for generating random number. And after that we assign the random number to “random_no” called integer. At line 14, this statement gives us the number to the console. so, you can see that at every time when we run the program it gives us the different number and make our code much easier than before.

Code

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <time.h>
4  using namespace std;
5  void wait_sec(int sec);
6  int main()
7  {
8     int sec=0;
9
10    cout << "After how many seconds you want to terminate the program : " ;
11    cin >> sec ;
12
13    cout << "Program starts ...!! " << "\n" ;
14    wait_sec( sec ) ;
15    cout << "Program terminated ...!! " << "\n";
16
17    return 0;
18 }
19
20 void wait_sec(int sec)
21 {
22          int Start_Time=time(0);
23          while( (time(0) - Start_Time) < sec )
24          {
25          }
26 }
```

Output for 1st run

```
After how many seconds you want to terminate the program : 2
Program starts ...!!
Program terminated ...!!

Process returned 0 (0x0)   execution time : 2.585 s
Press any key to continue.
```

Output for 2nd run

```
After how many seconds you want to terminate the program : 5
Program starts ...!!
Program terminated ...!!

Process returned 0 (0x0)   execution time : 5.169 s
Press any key to continue.
```

Output for 3rd run

```
After how many seconds you want to terminate the program : 7
Program starts ...!!
Program terminated ...!!

Process returned 0 (0x0)   execution time : 7.181 s
Press any key to continue.
```

Link:

Description

In above code we use **time(0)** to make **wait_sec( )** function. First this function asks you time in seconds for execution of program by that you can see whether time(0) function give you correct time or not. When you give time in seconds then present time will be stored in the “Start_time” named integer. Then at next line, this statement calculate that, **current_time**(continuous time) -(minus) **start_time**(stored time when program starts) is less than or greater than seconds(which user input). If less, than this loop will be continued but if greater, than the while loop will break and termination message shows on console and your program will terminate. On console, you can see that user inputed seconds is mostly equal to the program execution time.

8.3 Arrays

Simple Representations

In programming, what if we have to store a bulk of data of same type. For example if we have to store the marks of 10 students or may be more, first method is you can make 10 variables of same data types which is very painful method, another method is you can use Array. In C++ the syntax used to declare an array is given below

```
Datatype array_name[const_int_size];
```

An array is collection of data of same type and same name. For example

```
float marks[10];
```

Here, we declared an array, **marks**, of **float** type and **size** is 10. Meaning, it can hold 10 floating-point values.

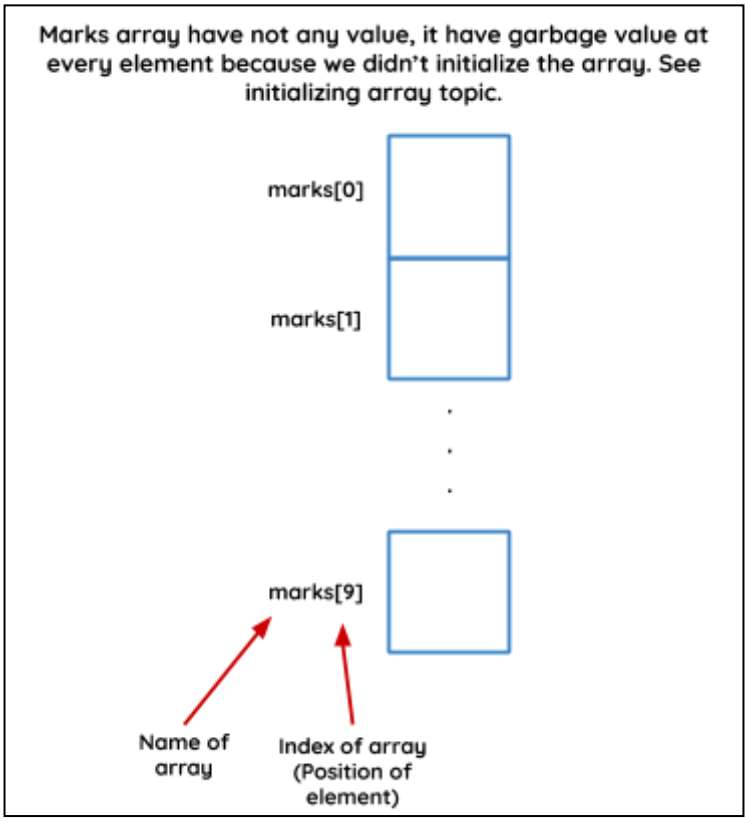
Types of Array

- Single Dimensional Array
- Multi Dimensional Array

In this chapter you are going to learn single dimensional array and the rest are in next chapters.

Accessing the Elements of Array

There are many methods to access the elements of array but in this chapter we are going to access it by indices of array. As you see above we declared an array **marks**. The first element is **marks[0]** , second element is **marks[1]** and so on.



Remember
In C++ the first index of array is always zero not 1, as you see above marks[0] is the first element and the last value will be at index size-1, like marks[9] is last value. We can think about these indices as the offset (how long you have to move further) from the base address which the array name is pointing towards.

Different Initializing methods

Using a Loop to Initialize the Array's Elements and Printing

Code

```
1  //Declaring an Array and Using a Loop to Initialize the Array's Elements with zero
2  #include<iostream>
3  using namespace std;
4  int main()
5  {
6          int marks[10];  //array declaration
7          for(int a=0; a<10; a++)
8          {
9              marks[a] = 0;  //using for loop to initialize the elements with zero
10         }
11         for(int b=0; b<10; b++)
12         {
13             cout << "[" << b << "] = " << marks[b] << endl;
14         }
15         return 0;
16 }
```

Link:

Output

```
[0] = 0
[1] = 0
[2] = 0
[3] = 0
[4] = 0
[5] = 0
[6] = 0
[7] = 0
[8] = 0
[9] = 0
```

Description

In this program an array named **marks** is declared with 10 elements at line 6. At line 9 in place of index, we pass variable **a** and it is replaced by its value when the loop iterates and all the elements declared zero. At line 13 we simply display the values of array again by using its indexes.

Initializing the elements of an array(with zero) with a declaration

Code

```
1  //Program A. Declaring an Array and Initialize the Array's
2  Elements with zero
3  #include<iostream>
4  using namespace std;
5  int main()
6  {
7          int marks[10] = {0};  //array declaration
8          for(int b=0; b<10; b++)
9          {
1          cout << "[" << b << "] = " << marks[b] << endl;
0          }
11         return 0;
12 }
```

```
//Program B. Declaring an Array without Size and Initializing
#include<iostream>
using namespace std;
int main()
{
        int marks[ ] = {55, 89, 54, 87, 100}; //array
declaration
        for(int b=0; b<5; b++)
        {
        cout << "[" << b << "] = " << marks[b] << endl;
        }
        return 0;
}
```

	Output A	Output B
	[0] = 0 [1] = 0 [2] = 0 [3] = 0 [4] = 0 [5] = 0 [6] = 0 [7] = 0 [8] = 0 [9] = 0	[0] = 55 [1] = 89 [2] = 54 [3] = 87 [4] = 100

Description A

This program is same as above but with a little change you see on line 6 array **marks** is declared and initialized all the elements with zero during deceleration

Remember

In C++ if you use any other number than zero during declaration it will not initialize all the elements with that number, only zero is the special number in C++ which initialize all the elements with zero.

Description B

In this program array **marks** is declared at line 6 without giving the size and initialized few elements of array. At line 7 we use **for** loop upto 5 iteration because elements in array are 5. We use 5 iteration to display elements because we know we initialize total 5 elements inside parenthesis **{ }**

Remember

In C++ if you not mention array size the compiler automatically detect the the size with it elements. As you see above we didn't mention the size of array and initialize 5 elements in array the compiler automatically detect the size of array is 5.

Passing Arrays to Function

Write a function that print the elements of array upto it's size

Code

```
1  //Passing an array to function and display
2  #include<iostream>
3  using namespace std;
4  void PrintArray(int A[ ], int Size)
5  {
6          for(int a=0; a<Size; a++)
7          {
8          cout << "[" << a << "] = " << A[a] << endl;
9          }
10 }
11 int main()
12 {
13         int size = 5;
14         int marks[size] = {55, 89, 54, 87, 100};  //array declaration
15         PrintArray(marks, size);
16         return 0;
17 }
18
```

Link:

Output

```
[0] = 55
[1] = 89
[2] = 54
[3] = 87
[4] = 100
```

Description

In this program function named **PrintArray** is created at line 4 and passed it an array and size of array, inside this function **for** loop is used which iterated up to less than **size** because last index of array is always one less than actual number. As you see at line 14 **size** variable is used in place to array size, it is another method to give size to an array.

8.3.1 Initializing Different Lists

Write a program which displays a menu and Initialize the array with the following different values, depending upon which option the user wants to test.

T1: Initialize with all array values to be zeroes {0, 0, 0, 0, 0, …….}
T2: {0, 1, 2, 3,4 ,5 …….. }
T3: {1, 2, 3, 4, 5, …….}
T4: {2, 4, 6, 8, 10, …….. }
T5: {10, 15, 20, 25, 30, 35, …….} Starting from 10 with 5 more than the previous value.
T6: {T*0, T*1, T*2, T*3, ……..} where T is entered by the user
T7: {T*1, T*2, T*3, T*4, ……..} where T is entered by the user
T8: {2, 3, 5, 7, 11, 13, 17, 19, …….} Save all the prime numbers
T9: {Assigning random values}
T10: {Assigning random Even numbers}
T11: {Assigning random numbers within a range}
*__T12__: {Assigning random prime numbers between first 1000 prime numbers}

```
1  //Declaration and Initialization of Array
2  #include<iostream>
3  #include<conio.h>
4  using namespace std;
5  void T1(int A[ ], int size);
6  void T2(int A[ ], int size);
7  void T3(int A[ ], int size);
8  void T4(int A[ ], int size);
9  void T5(int A[ ], int size);
10 void T6(int A[ ], int size);
11 void T7(int A[ ], int size);
12 bool IsPrime(int N);
13 void T8(int A[ ], int size);
14 void T9(int A[ ], int size);
15 void T10(int A[ ], int size);
16 void T11(int A[ ], int size);
17 void T12(int A[ ], int size);
18 void Menu();
19 void PrintArray(int A[ ], int Size);
20 int main()
21 {
22         const int size = 10;
23         int Choice; bool Exit = false;
24         int arrayT[size];  //array declaration
25
26         do
27         {
28                 Menu();
29                 cin >> Choice;
30                 switch (Choice)
31                 {
32                 case 1:
33                         T1(arrayT, size);
34                         break;
35                 case 2:
36                         T2(arrayT, size);
37                         break;
38                 case 3:
39                         T2(arrayT, size);
40                         break;
41                 case 4:
42                         T2(arrayT, size);
43                         break;
44                 case 5:
45                         T2(arrayT, size);
46                         break;
47                 case 6:
48                         T2(arrayT, size);
49                         break;
50                 case 7:
51                         T2(arrayT, size);
52                         break;
53                 case 8:
54                         T2(arrayT, size);
55                         break;
56                 case 9:
57                         T2(arrayT, size);
58                         break;
59                 case 10:
60                         T2(arrayT, size);
```

```
61                              break;
62                      case 0:
63                              Exit = true;
64                              break;
65                      }
66                      PrintArray(arrayT, size);
67                      cout << "\n\nPress any key to continue...\n\n";
68                      _getch();
69              }
70              while (Exit != true);
71
72              return 0;
73      }
74      void Menu()
75      {
76              system("cls");
77              cout <<"1:    {0, 0, 0, 0, 0,...}" << endl
78                      << "2:    {0, 1, 2, 3, 4,       ...}" << endl
79                      << "3:    {1, 2, 3, 4, 5,       ...}" << endl
80                      << "4:    {2, 4, 6, 8, 10,      ...}" << endl
81                      << "5:    {10, 15, 20, 25,      ...}" << endl
82                      << "6:    {T*0, T*1, T*2, T*3,...} where T is entered by the user" << endl
83                      << "7:    {T*1, T*2, T*3, T*4,...} where T is entered by the user" << endl
84                      << "8:    {2, 3, 5, 7, 11,      ...} Save all the prime numbers" << endl
85                      << "9:    {Assigning random values}" << endl
86                      << "10:   {Assigning random Even numbers}" << endl
87                      << "11:   {Assigning random numbers within a range}" << endl
88                      << "12:   {Assigning random prime numbers between first 1000 prime numbers}" << endl
89                      << "0:    For Exit"<<endl
90                      << "\n\nSelection Option: ";
91      }
92
93      void PrintArray(int A[ ], int Size)
94      {
95              for (int ai = 0; ai<Size; ai++)
96              {
97                      cout << "[" << ai << "] = " << A[ai] << endl;
98              }
99      }
100     void T1(int A[ ], int size)
101     {
102             for (int ai = 0; ai<size; ai++)
103                     A[ai] = 0;// we have done this before do it yourself
104     }
105     void T2(int A[ ], int size)
106     {
107             for (int a = 0; a<size; a++)
108                     A[a] = a;
109     }
110     void T3(int A[ ], int size)
111     {
112             for (int a = 0; a<size; a++)
113                     A[a] = a + 1;
114     }
115     void T4(int A[ ], int size)
116     {
117             for (int a = 0; a<size; a++)
118             {
119                     A[a] = (a + 1) * 2;
120             }
```

```
121 }
122 void T5(int A[ ], int size)
123 {
124         for (int a = 0; a<size; a++)
125         {
126                 A[a] = (5 * a) + 10;
127         }
128 }
129 void T6(int A[ ], int size, int num)
130 {
131         for (int a = 0; a<size; a++)
132         {
133                 A[a] = num * a;
134         }
135 }
136 void T7(int A[ ], int size, int num)
137 {
138         for (int a = 0; a<size; a++)
139         {
140                 A[a] = num * (a + 1);
141         }
142 }
143 bool IsPrime(int N)
144 {
145         if (N <= 1)
146         {
147                 return false;
148         }
149         for (int d = 2; d <= N / 2; d++)
150         {
151                 if (N%d == 0)
152                 {
153                         return false;
154                 }
155         }
156         return true;
157 }
158
159 void T8(int A[ ], int size)
160 {
161         int k = 0;
162         for (int i = 0; i<size; i++)
163         {
164                 for (int j = k; true; j++)
165                 {
166                         if (IsPrime(j))
167                         {
168                                 A[i] = j;
169                                 k = j + 1;
170                                 break;
171                         }
172                 }
173         }
174 }
175 void T9(int A[ ], int size)
176 {
177         for (int i = 0; i<size; i++)
178         {
179                 A[i] = rand();
180         }
```

```
}
void T10(int A[ ], int size)
{
        int z;
        for (int i = 0; i<size; i++)
        {
                while (1)
                {
                        z = rand();
                        if (z % 2 == 0)
                        {
                                A[i] = z;
                                break;
                        }
                }
        }
}
void T11(int A[ ], int size, int start, int end)
{
        int z, rangeSize = start-end+1;
        for (int i = 0; i<size; i++)
        {
                z = rand()%rangeSize + start;
                A[i] = z;
        }
}
```

Here are the possible selections of initialization which will initialize the array accordinly.

T1

```
[0] = 0
[1] = 0
[2] = 0
[3] = 0
[4] = 0
[5] = 0
[6] = 0
[7] = 0
[8] = 0
[9] = 0
```

T2

```
[0] = 0
[1] = 1
[2] = 2
[3] = 3
[4] = 4
[5] = 5
[6] = 6
[7] = 7
[8] = 8
[9] = 9
```

T3

```
[0] = 1
[1] = 2
[2] = 3
[3] = 4
[4] = 5
[5] = 6
[6] = 7
[7] = 8
[8] = 9
[9] = 10
```

T4

```
[0] = 10
[1] = 15
[2] = 20
[3] = 25
[4] = 30
[5] = 35
[6] = 40
[7] = 45
[8] = 50
[9] = 55
```

T5

```
enter any number of your choice: 65
[0] = 0
[1] = 65
[2] = 130
[3] = 195
[4] = 260
[5] = 325
[6] = 390
[7] = 455
[8] = 520
[9] = 585
```

T6

```
enter any number of your choice: 15
[0] = 15
[1] = 30
[2] = 45
[3] = 60
[4] = 75
[5] = 90
[6] = 105
[7] = 120
[8] = 135
[9] = 150
```

T7

```
[0] = 2
[1] = 3
[2] = 5
[3] = 7
[4] = 11
[5] = 13
[6] = 17
[7] = 19
[8] = 23
[9] = 29
```

T8

```
[0] = 41
[1] = 18467
[2] = 6334
[3] = 26500
[4] = 19169
[5] = 15724
[6] = 11478
[7] = 29358
[8] = 26962
[9] = 24464
```

T9

```
[0] = 6334
[1] = 26500
[2] = 15724
[3] = 11478
[4] = 29358
[5] = 26962
[6] = 24464
[7] = 11942
[8] = 5436
[9] = 14604
```

and so on.

Description

In this program two function is created first **T1,** it is used to initialize the array and second function **PrintArray**

is used to print the values of array as you see in output. In **int main** these two functions is called and passed them array and size.

Description
In this program we have to initialize all the elements of array with whole numbers up to size, so that why at line 25 iteration variable **a** is used to store, when the loop iterates from **0** to **size** these values store in array one by one. As you see the output, array stores whole numbers up to 9.

Description
In this program we have to initialize all the elements of array with integer numbers up to size, so that why at line **a + 1** is used to store, when the loop iterates from **0** to **size** these values store in array with addition one because first iteration starts from zero. As you see the output, array stores integers 10.

Description
In this program we have to initialize all the elements of array with even numbers up to size, so that why at line **(a + 1) * 2** is used to store, when the loop iterates from **0** to **size** these values store in array with addition one and multiply by two because first iteration starts from zero. As you see the output, array stores even integers from 2 to 20.

Description
In this program we have to initialize first element with 10 and rest are with addition 5 up to size, so that why at line **(5 * a) + 10** is used to store, when the loop iterates from **0** to **size** these values store in array with addition ten and multiply **a** with five because first iteration starts from zero. As you see the output above.

Description
In this program we have take input(number) from user and multiply it with whole numbers one by one, so that at line **num * a** is used to store in array, when the loop iterates from **0** to **size** these user entered number(**num**) multiply with **a**(loop iteration) and store in array. As you see the output above.

Description
In this program we have take input(number) from user and multiply it with whole numbers one by one, so that at line 25 **num * a** is used to store in array, when the loop iterates from **0** to **size** these user entered number(**num**) multiply with **a**(loop iteration) and store in array. As you see the output above.

Description

Description
In this program at line 27 we are assigning random values to every element(index) of array **arrayT9.** The rest of the code are same as previous programs.

Description
In this program we are assigning even random values to every element(index) of array **arrayT10.** Inside for loop at line 24 an infinite loop is applied which iterate until z do not store even random number which checked using condition(**z%2 == 0**) at line 27, when the even random value is stored in array the infinite while loop breaks. This process is repeated until for loop is finished.

Remember
In C++ when array is passed to function than it is actually passed by reference, which means if you change something in array in function it is automatically changed in int main too. ~~Although this is a~~

~~*lie, but for the time being believe in this.*~~

8.3.2 Bubble Sort

Before starting Bubble Sort Algorithm, first we have to learn about Swap Algorithm.

8.3.2.1 Swap

In this algorithm we change value of both variables. Firstly we declare a new variable **t** and assign value of first variable **a** to this newly declared variable **t** and then assign value of second variable **b** to first variable **a** and then assign value of newly declared variable **t** to second variable **b**. Thus without losing any value we exchange values of both variables **a** and **b**.

Code

```
1   #include<iostream>
2   using namespace std;
3   void swap(int &a,int &b);
4   int main()
5   {
6           int V1=20,V2=30;
7           cout<<"before swap :  "<<"V1= "<<V1<<"  V2="<<V2<<endl;
8           swap(V1,V2);
9           cout<<"after swap :  "<<"V1= "<<V1<<"  V2="<<V2<<endl;
10          return 0;
11  }
12  void swap(int &a,int &b)
13  {
14          int t=a;
15          a=b;
16          b=t;
17  }
```

Output

```
before swap :  V1= 20  V2=30
after swap :  V1= 30  V2=20
```

Link

Description

In above code at line 1 and 2 we have used some C++ libraries. At line 3 we have declared function **swap** and from line 4 to 11 **int main( )** of this program is written. At line 6 we have declared two variables **V1** and **V2**, and assigned them values 20 and 30 respectively. At line 7 we have displayed values of these variables on console and at line 8 we have passed these variables to **swap( )** which exchanges their values. At line 9 we displays the exchanged values of variables **V1** and **V2** on console and at line 10 we terminates the program by **return 0**. At line 12 we have placed code of **swap( )**, whose algorithm is described in initial description of this topic.

8.3.2.2 Bubble Sort

In this topic we will describe how to sort in ascending order. Let's find methods to sort numbers. One way to sort a series of numbers is by using **Bubble Sort Method**. This method is related to bubbles formed while boiling water. As we know while boiling water, the molecules whose kinetic energy is high is converted to bubbles first. Similarly in this method the biggest number is placed first at last position. In this method first we compare two numbers from starting positions. If first number is bigger than second numbers we exchange their positions (can be done by using **swap( )** function) else it is confirmed that first number is smaller than second number so we do nothing. We have to do comparison between these till total numbers -1 size because we have to compare between two numbers, if we do this comparison till total numbers then at last comparison we will have only one number and we cannot do comparison. When we have done with this comparison first time till total numbers -1 time then we have placed biggest number at last position. We have to repeat the same

steps for other numbers i.e. total numbers -2 because when we have placed second smallest number at its right position than smallest number is already placed at its right position.

Code of Bubble Sort Algorithm is given below:

Code

```
#include<iostream>
using namespace std;
int init(int A[ ],int a_size);
void swap(int &a,int &b);
void bubble_up(int A[ ],int a_size);
void sort(int A[ ],int a_size);
void print_array(int A[ ],int size_of_array);
int main()
{
        int A[5];
        int a_size=5;
        init(A,a_size);
        cout <<"before sort : ";
        print_array(A,a_size);
        cout <<endl;
        sort(A,a_size);
        cout <<"after sort : ";
        print_array(A,a_size);
        cout <<endl;
        return 0;
}
void swap(int &a,int &b)
{
        int t=a;
        a=b;
        b=t;
}
void bubble_up(int A[ ],int a_size)
{
        for(int i=0;i<a_size-1;i++)
        {
           if(A[i]>A[i+1])
               swap(A[i],A[i+1]);
        }
}
void sort(int A[ ],int a_size)
{
        for(int i=0;i<a_size;i++)
             bubble_up(A,a_size);
}
void print_array(int A[ ],int size_of_array)
{
        for(int i=0;i<size_of_array;i++)
        {
        cout <<A[i]<<" ";
        }
}
int init(int A[ ],int a_size)
{
        cout<<"enter "<<a_size<<" numbers : ";
        for(int i=0;i<a_size;i++)
        {
        cin>>A[i];
        }
}
```

Output

```
enter 5 numbers : 5 3 6 9 1
before sort : 5 3 6 9 1
after sort : 1 3 5 6 9
```

Link:
Description
In above code from line 1 to 7 we have defined some library functions and some prototypes of function we will use in this function. From line 8 to 21 **int main( )** is written. At line 10 we initialize an array **A** of 5 elements. At line 11 we declared an integer **a_size** and initialize its value by 5 which we use as size of array when passing array **A** to a function. Ata line 12 we call **init( )** which initialize array by getting inputs from users. At line 12 we display a message at console and at line 13 we call **print_array** which displays value of array on console. At line 16 we call **sort( )** which sorts array **A** using Bubble Sort Algorithm, as described above. At line 17 we display a message on console and at line 8 we again call **print_array** which at this time displays sorted array on console and then we termite our program by **return 0**.

Exercise
Your task is to initialize array with random values using rand( ), by making changing in init( ) function.

8.3.4 Selection Sort

In this type of sorting we check the smallest number from given numbers and place that smallest number at first position i.e. swap the position of smallest number with first number. Then we will find the second minimum number and place it at second position(here again we call swap function). We will continue this process till total numbers -1 because when we place second biggest number at its correct position then biggest number is already placed at its right position. For this we need a **Selection_Sort** function which will get array of numbers and its size and then it will call further call **swap** function after determining index of array where minimum number is saved. We will also need **print_array, init** functions which we already used in Bubble Sort algorithm for same purpose.

Code

```
#include<iostream>
using namespace std;
void print_array(int A[ ],int size_of_array);
void swap(int &a,int &b);
int init(int A[ ],int a_size);
void Selection_Sort(int A[ ],int size);
int Find_min_Inx_Range(int A[ ],int si,int ei);
int main ()
{
        int A[5];
        int a_size=5;
        init(A,a_size);
        cout <<"before sort : ";
        print_array(A,a_size);
        cout <<endl;
        Selection_Sort(A,a_size);
        cout <<"after sort : ";
        print_array(A,a_size);
        cout <<endl;
        return 0;
}
void print_array(int A[ ],int size_of_array)
{
        for(int i=0;i<size_of_array;i++)
        {
        cout <<A[i]<<" ";
        }
}
int init(int A[ ],int a_size)
{
        cout<<"enter "<<a_size<<" numbers : ";
        for(int i=0;i<a_size;i++)
        {
        cin>>A[i];
        }
}
void Selection_Sort(int A[ ],int size)
{
        int min;
        for(int i=0;i<size-1;i++)
        {
            min=Find_min_Inx_Range(A,i,size-1);
            swap(A[i],A[min]);
        }
}
void swap(int &a,int &b)
{
        int t=a;
        a=b;
        b=t;
}
int Find_min_Inx_Range(int A[ ],int si,int ei)
{
        int min=A[si];
        int min_index=si;
        for(int i=si;i<=ei;i++)
        {
        if(min>A[i])
        {
        min=A[i];
        min_index=i;
        }
        }
        return min_index;
}
```

Output

```
enter 5 numbers : 3 2 9 6 1
before sort : 3 2 9 6 1
after sort : 1 2 3 6 9
```

Link:

Description

In above code from line 1 to 7 we have defined some c++ library functions and some prototypes of function which we will use in our code. From line 8 to 21 **int main()** of this program is described. In **int main()** at line 10 we have declared an array of size 10 and line 11 we declared **a_size** which represents size of array. At line 11 we call **init()** which initializes this array. At line 13 we print a message on console and at line 14 we call **print_array** which print numbers stored in array at console. At line 16 we call **Selection_Sort()** which sorts array according to above described algorithm using **for** loop,**Find_min_Inx_Range( )** and **swap( )**. And at line 17 we display a message on console and at line 19 we again call **print_array** which displays sorted array on console.

8.3.5 Finding Extreme Values

Write and test the following function that returns through its reference parameters both the maximum and the minimum values stored in an array:
void getExtremes(int & min, int & max, int a[ ], int n);

Code

```
1  /*
2  Declaration, Random Initialization and finding minimum
3  and maximum value from an Array
4  */
5  #include<iostream>
6  #include<stdlib.h>
7  using namespace std;
8  void init(int A[ ], int size);
9  int minValue(int A[ ], int size);
10 int maxValue(int A[ ], int size);
11 void getExtremes(int & MIN , int & MAX, int a[ ], int size);
12 void PrintArray(int A[ ], int size);
13 int main()
14 {
15         int size;
16         cout << "enter array size: ";
17         cin >> size;
18         int A[size];
19         init(A, size);
20         PrintArray(A, size);
21         int min=0, max=0;
22         getExtremes(min, max, A, size);
23         cout << endl << endl << "minimum value is:" << min << endl;
24         cout << "maximum value is:" << max << endl;
25         return 0;
26 }
27 void init(int A[ ], int size)
28 {
29         for(int i=0; i<size; i++)
30         {
31         A[i] = rand()%100;
32         }
33 }
34 int minValue(int A[ ], int size)
35 {
```

```
36              int min = A[0];
37              for(int i=1; i<size; i++)
38              {
39              if(A[i]<min)
40              {
41              min=A[i];
42              }
43              }
44              return min;
45  }
46  int maxValue(int A[ ], int size)
47  {
48              int max = A[0];
49              for(int i=1; i<size; i++)
50              {
51              if(A[i]>max)
52              {
53              max=A[i];
54              }
55              }
56              return max;
57  }
58  void getExtremes(int & MIN , int & MAX, int a[ ], int size)
59  {
60              MIN = minValue(a,size);
61              MAX = maxValue(a,size);
62  }
63  void PrintArray(int A[ ], int size)
64  {
65              for(int b=0; b<size; b++)
66              {
67              cout <<endl<< "["<<b<<"] = " << A[b];
68              }
69  }
```

Output

```
enter array size: 10

[0] = 41
[1] = 67
[2] = 34
[3] = 0
[4] = 69
[5] = 24
[6] = 78
[7] = 58
[8] = 62
[9] = 64

minimum value is:0
maximum value is:78
```

Link:
Description

8.3.6 Removing a number from an array

Write and test the following function that removes an item from an array
void remove(float a[ ], int& n, int i);

The function removes a[i] by shifting all the elements above that position are down and decrementing n.

Actually, In this algorithm we just have to remove a number from an array. Now firstly to initialize an array we will take size of an array from the user and then we will take number of inputs in an array at size times. Then we will ask user to enter the number which he/she wants to remove. Then we will check that number from the array and if the number found we will remove it by just assigning the value at the current position of that number which is very next to it and will do this same thing at size-current_position time and at last decrease one from the size.

After that we will again check the array and again if the number is found then we will do the same steps as above to remove the number from array and we will check array at the size times.

Code

```
1   #include <iostream>
2   using namespace std;
3   void init(float a[ ],int n);
4   void ShiftArray(float a[ ],int& n,int x);
5   void RemoveNumber(float a[ ], int& n, int i);
6   void PrintArray(float a[ ],int n);
7   int main()
8   {
9           int n=0;
10          float num=0;
11          cout<<"Enter size of array:";
12          cin>>n;
13          float array[n];
14          init(array,n);
15          cout<<"enter number to remove";
16          cin>>num;
17          RemoveNumber(array,n,num)
18  m);
19          PrintArray(array,n);
20  }
21  void init(float a[ ], int n)
22  {
23          cout<<"Enter Size Values: ";
24          for(int i=0;i<n;i++)
25          {
26          cin>>a[i];
27          }
28  }
29  void ShiftArray(float a[ ],int& n,int x)
30  {
31          for(int i=x;i<n-1;i++)
32          {
33          a[i]=a[i+1];
34          }
35          n--;
36  }
37  void RemoveNumber(float a[ ], int & n, int i)
38  {
39          for(int x=0;x<n;x++)
40          {
41          if(a[x]==i)
42          {
43          ShiftArray(a,n,x);
44          }
45          }
46  }
47  void PrintArray(float a[ ],int n)
48  {
49          cout<<"\nArray after removing:";
50          for(int i=0;i<n;i++)
51          {
52          cout<<a[i];
53          }
    }
```

Output

```
Enter size of array:5
Enter values of array:
1
2
3
4
5
enter number to remove:3

Array after removing:1245
```

Description

In above code from line 1 to 6 we have defined some library functions and some prototypes of function we will use in this program and from 7 to 19 **int main( )** is written. In **int main( )** at line 14 we have call a function **Init** and send an **array** and **n** as a parameter to it. This function initializes the array by the numbers entered by the user. And then at line 16 we have ask user to enter the number which he wants to remove from array. Now to remove a number from the array. we have call a function **RemoveNumber** at line 17 and send **array, n** and that number **num** as parameter to this function. Now this function will check that if any of the array element match with that number then it will call a further function **ShiftArray** at line 42 and send **a,n** and **x** as a parameter to it. This function will assign the value at current position which is very next to its address at line 32 at n-1 times starting from x. And at last decreases one from the n.

8.3.7 Rotating inside an array

Write and test the following function
void rotate(int a[ ], int n, int k);
The function “rotates” the first n elements of the array a, k positions to the right (or –k positions to the left if k is negative). The last k elements are “wrapped” around to the beginning of the array. For example, the call rotate(a,8,3) would transform the array {22,33,44,55,66,77,88,99} into {77,88,99,22,33,44,55,66}. The call rotate(a,8,-5)would have the same effect.

Code

```
#include <iostream>
using namespace std;
void Print(int Array[ ],int Size)//It will Print Array Values on the screen
{
	for(int i=0;i<Size;i++)
	{
	   cout<<Array[i]<<" ";
	}
	cout<<endl;
}
void LeftRotate(int Array[ ],int Size)//It Will rotate left the array values
{
	int temp=Array[0];
	for(int i=0;i<Size-1;i++)
	{
	    Array[i]=Array[i+1];
	}
	Array[Size-1]=temp;
}
void RightRotate(int Array[ ],int Size)//It will rotate right the values of array
{
	int temp=Array[Size-1];
	for(int i=0;i<Size-1;i++)
	{
	    Array[Size-1-i]=Array[Size-2-i];
	}
	Array[0]=temp;
}
void MainRotator(int Array[ ],int Size,int K)//It is the main function in which we mainly rote values
{
	if(K>=0)
	{
	   for(int i=0;i<K;i++)
	   {
	      RightRotate(Array,Size);
	   }
	}
	else
	{
	  for(int i=K;i<0;i++)
	  {
	    LeftRotate(Array,Size);
	  }
	}
	Print(Array,Size);
}
int main()
{
	int Size=0;
	cout<<"enter size: ";
	cin>>Size;
	int Array[Size];
	cout<<"enter "<<Size<<" values: ";
	for(int i=0;i<Size;i++)
	{
	  cin>>Array[i];
	}
	int K=0;
	cout<<"enter how many times YOu want to Rotate :"<<endl;
	cout<<"enter Positive value for rotating right and negative to rotate   left: ";
	cin>>K;
	MainRotator(Array,Size,K);
	return 0;
}
```

Link

Description

In it we firstly enter values from user then we enter our main function **MainRotator** in it there are further two more functions first the **left rotate** and **right rotate** these function rotate our array left or right once in **left rotate** function we save first value in **temp** then we move all our values using **for loop** to left and at last assign **temp** to the last value on **array** and in the same but opposite way to move right in **right rotate** function. Then at last came our **main rotate** function in it we rotate the array using **MainRotator** function **k** times or according to the value entered of **k**. And at last we print the result on the screen using the **print** function.

Output

```
enter size: 5
enter 5 values: 1 2 3 4 5
enter how many times YOu want to Rotate :
enter Positive value for rotating right and negative to rotate left: 3
3 4 5 1 2
```

8.3.8 Appending an Array

Write and test the following function
void append(int a[ ], int & m, int b[ ], int n);
The function appends the first n elements of the array b onto the end of the first m elements of the array a. It assumes that a has room for at least m + n elements. Note that it must also change the size of the first array to m+n. For example, if a is {22,33,44,55,66,77,88,99} and b is {20,30,40,50,60,70,80,90} then the call append(a,5,b,3) would transform a into {22,33,44,55,66,20,30,40}. Note that b is left unchanged and only n elements of a are changed.

Code

```
#include <iostream>
#include<stdlib.h>
using namespace std;
void Print_Array(int A[ ],int Size)
{
        for(int i=0;i<Size;i++)
        {
          cout<<"A"<<"["<<i<<"]\t";
        }
        cout<<endl;
        for(int i=0;i<Size;i++)
        {
          cout<<A[i]<<" \t";
        }
}
void Initialize(int A[ ],int size,int r)
{
        for(int i=0;i<size;i++)
        {
          A[i]=rand ( )%r;
        }
}
void Append(int A[ ],int B[ ],int x,int y)
{
        int size;
        size=x+y;
        int C[size];
        for(int j=0;j<x;j++)
        {
          C[j]=A[j];
        }
    for(int i=0;i<y;i++)
        {
          C[x+i]=B[i];
        }
        Print_Array(C,size);
}
int main()
{
        int size,n;
        cout<<"Enter Array size: ";
        cin>>size;
        int A[size],B[size];
        cout<<"Enter range: ";
        cin>>n;
        Initialize(A,size,n);
        Initialize(B,size,n);
        Print_Array(A,size);
        cout<<endl;
        Print_Array(B,size);
        cout<<endl;
        int x,y;
        cout<<"Enter any two values: ";
        cin>>x>>y;
        Append(A,B,x,y);
        cout<<endl;
        return 0;
}
```

Output

```
Enter Array size: 11
Enter range: 11
A[0]    A[1]    A[2]    A[3]    A[4]    A[5]    A[6]    A[7]    A[8]    A[9]    A[10]
8       9       9       1       7       5       5       10      1       0       7
A[0]    A[1]    A[2]    A[3]    A[4]    A[5]    A[6]    A[7]    A[8]    A[9]    A[10]
7       5       8       6       7       3       7       9       2       7       7
Enter any two values: 5 3
A[0]    A[1]    A[2]    A[3]    A[4]    A[5]    A[6]    A[7]
8       9       9       1       7       7       5       8
```

Description

In this code first we have to make 2 arrays. Than we will pass the arrays one by one to a function named **initialize.** This function will take 3 parameters. First one is array and the second one is array size. The third one is the range of numbers in that array. After returning from this function both arrays will be initialized with some random values. From line number 52 to 54 we are asking the user to enter 2 values (x and y). We are actually asking the user that how many numbers he wants to get from Array1 and Array2 to display it on Array3. Than we will pass both of these values and oth Arrays to the function named **Append.** In this function first we will make an array of size x+y. After that we will create a for loop which will iterate up to x and copy the first x elements of Array1 to Array3. Than we will create another for loop which will iterate up to y but this time we will copy the elements of Array2 to Array3 and we will copy the values to the next index where we left at the first loop. That why we have written **C[x+i]=B[i]** at line 34.

8.3.9 Frequency of an element Inside an array, Mode/Median of Population

Write and test the function
int frequency(float a[ ], int n, int x);
This function counts the number of times the item x appears among the first n elements of the array a and returns that count as the frequency of x in a.

Finding the Mode

Code

```
1   #include <iostream>
2   using namespace std;
3
4   void init(float a[ ],int n);
5   int frequency(float a[ ], int n, int x);
6   int FindingMode(float a[ ],int n);
7   int main()
8   {
9     int n=0;
10    int value=0;
11    int count=0;
12    int mode_value=0;
13    cout<<"Enter size of array:";
14    cin>>n;
15    float array[n];
16    init(array,n);
17    mode_value=FindingMode(array,n);
18    if(mode_value==1)
19          cout<<"None";
20    else
21          cout<<"Mode of the array is:"<<mode_value;
22  }
23  void init(float a[ ], int n)
24  {
25          cout<<"Enter values of array:";
26          for(int i=0;i<n;i++)
```

```
27          {
28          cin>>a[i];
29          }
30      }
31      int frequency(float a[ ], int n, int x)
32      {
33          int count=0;
34          for(int i=0;i<n;i++)
35          {
36          if(a[i]==x)
37          {
38          count++;
39          }
40          }
41          return count;
42      }
43      int FindingMode(float a[ ],int n)
44      {
45          int mode=0;
46          int count=0;
47          int count2=0;
48          for(int i=0;i<n;i++)
49          {
50          count=frequency(a,n,a[i]);
51          if(count>=count2)
52          {
53          mode=a[i];
54          count2=count;
55          }
56          }
57          if(count2==1)
58          {
59          return count2;
60          }
61          return mode;
62      }
```

Output

```
Enter size of array:5
Enter values of array:
1
2
3
2
2

Mode of the array is:2
```

Description

Finding the most **repetitive number** from the given data is simply known as **Mode**. Now in this program we he to find the mode of the given data in an array. Hence for that in code we have make a function **FindindMode** from line 43 to 61 and send **a[ ](array)** and **n(size)** as a parameter to that function. This function checks the frequency of every element of an array and stores it in a variable, Now to for checking the frequency he have further called a function **frequency** at line 50 and send **a[ ]** , **n** and element of array **a[i]** as parameter and as a result this function return the frequency in a variable **count**, and if the **count** greater than or equal to the **count2** than **mode** will be equal to the **a[i]** at line 53 and **count2** will be equal to **count** at line 54 and the same thing will happens for **n** times in loop. Now if the **count2** will equal to 1 than it means there will be repetition in the array's element and function will return **count2** to the **main** ,but if it is not equal to 1 than it will return the number which is in the variable **mode.**

Finding the Median

Code

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

void init(float a[ ],int n);
void swap(float& a,int& b);
void bubble_up(float A[ ],int a_size);
void sort(float A[ ],int a_size);
void PrintArray(float a[ ],int n);
int FindMedian(float a[ ], int n);
int main()
{
        int n=0;
        cout<<"Enter size of array:";
        cin>>size;
        float array[n];
        init(array,n);
        PrintArray(array,n);
        cout<<"\nMedian of array is:"<<;
}
int FindMedian1(float a[ ], int n);
{
   Sort(a, n);                  // This will change the original Array
   return a[n/2];
}

int LessThanFrequency(int a[ ], int n, int T)
{
   int count = 0;
   for(int i=0; i<n; i++)
   {
     if(a[i]<T) count++;
   }
  return count;
}

int FindMedian2(float a[ ], int n);
{
    // This assumes that all the elements in an array are unique or the Median doesn't repeat.
    for(int i=0;i<n; i++)
    {
       if(LessThanFrequency(a, n, a[i]) == (n+1)/2-1 )   // This algo will use the array for read-Only purpose.
            return a[i];
    }
}

void init(float a[ ], int n)
{
        cout<<"Enter values of array:";
        for(int i=0;i<n;i++)
        {
        cin>>a[i];
        }
}

void PrintArray(float a[ ],int n)
{
        for(int i=0;i<size;i++)
        {
                cout<<a[i];
        }
}
```

Output

```
Enter size of array:5

Enter values of array:
1
2
3
4
5

12345

Median of array is:3
```

Description

Finding the Middle value from the **sorted data** is simply knows as **Median**. Now in this program we have to find the median of the data given an array. The data is entered by the user so it may be not sorted. Hence for that we will first sort our data and for sorting, In code we have already makes a function **sort** which will sort the given data in decreased numerical order itself by calling some further function as explained in 8.3.2.2and send **array** and **n** as a parameter to it on line 16. Now as we have to find the middle value of this function now for that will fill just ……...

8.3.10 Is the given date symmetric

Write and test the following function:
bool isSymmetric(int a[ ], int n);
The function returns true if and only if the array obtained by reversing the first n elements is the same as the original array. For example, if a is {22,33,44,55,44,33,22} then the call isSymmetric(a,7) would return true, but the call isSymmetric(a,4) would return false. Warning: The function should leave the array unchanged.

Code

```cpp
#include <iostream>
using namespace std;
bool IsSymmetric(int Array[ ],int Size)//it will check if the function is symmetric
{
	for(int i=0;i<Size/2;i++)
	{
	if(Array[i]!=Array[Size-i-1])
	 {
		return false;
	  }
	}
	return true;
}
int main()
{
	int Size=0;
	cout<<"enter size: ";
	cin>>Size;
	int Array[Size];
	cout<<"enter "<<Size<<" values: ";
	for(int i=0;i<Size;i++)
	{
	cin>>Array[i];
	}
	if(IsSymmetric(Array,Size))
	{
	cout<<"Array is Symmetric"<<endl;
	}
	else
	{
	cout<<"Array is not Symmetric"<<endl;
	}
	return 0;
}
```

Link

Output

```
enter size: 5
enter 5 values: 1 2 3 2 1
Array is Symmetric
```

```
enter size: 6
enter 6 values: 1 2 3 4 3 2
Array is not Symmetric
```

Description

We have to write a program which will enter numbers from user and store them in an **array** and tell whether the data enter by user is symmetric or not in other words if we read from both side left or right the data will remain same as for e.g 121. In it firstly we take all inputs from user in an **array** then we pass that **array** to our main function **IsSymmetric** Which will return **true** or **false** if the array or entered values are **symmetric** then **true else false**. Now we start designing that function as we know the **symmetric** number is same when read from both sides so we use a **for loop** in which we start from one end and goto the center of array checking that the values on the both end or on the **i** and **size -i -1** indexes are same or not if they all are same then it return **true** else it return **false**. Then according to the function result we gave our final **output** on the screen.

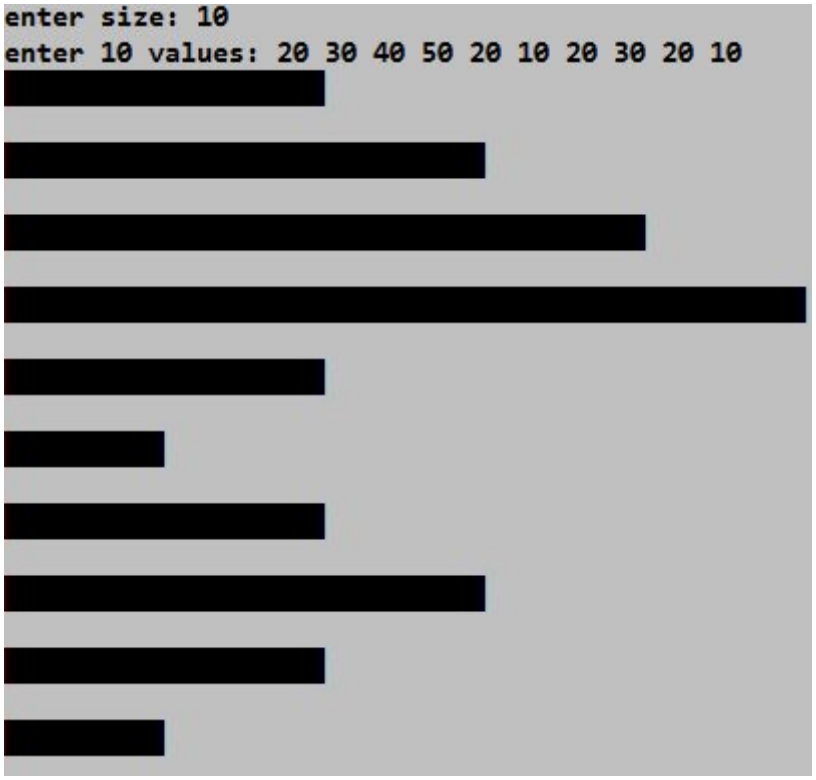
8.3.11 Plotting the histogram

Code

```cpp
#include <iostream>
using namespace std;
void PrintingHistogram(int Array[ ],int Size,char Symbol)//it will print histogram
{
	for(int i=0;i<Size;i++)
	{
	for(int x=0;x<Array[i];x++)
	{
	cout<<Symbol;
	}
	cout<<endl<<endl;
	}
}
int main()
{
	int Size=0;
	cout<<"enter size: ";
	cin>>Size;
	int Array[Size];
	cout<<"enter "<<Size<<" values: ";
	for(int i=0;i<Size;i++)
	{
	cin>>Array[i];
	}
	char Symbol=-37;  // the ascii value of filled character
	PrintingHistogram(Array,Size,Symbol);
	return 0;
}
```

Link

Output



Description

In this problem we have to write a program which will enter inputs from user and make a histogram graph of it so firstly we have have to enter values then we enter in the **PrintingHistogram** in it we use a **for loop** of **size** of array and an inner **for loop** which will print the symbol in screen in a row until the value enter by the user then a double **endline** to move toward the 2nd next line in this way at the end of loop our histogram will be printed.

8.4 String (Character Arrays)

Just like the integer array we also have character array in which we can store multiple characters in our character array. There are three basic ways to fill up our character array. First is the same as we fill up our integer array but the character array must be initialized and it is initialized by null character in coding you will write it as
Char array{ }={"\0"} you can see the code given below as an examples.

Type 1 Using for loop

Example

You have to check whether the user entered data is Palindrome or not in other words you have to check that the data is Symmetric or not.

Code

```
#include <iostream>
using namespace std;
int LenghtOfArray(char Array[ ],int Size);//It will Return the length of Array
bool IsPalindrome(char Array[ ],int Size);//It will check if the array is palindrome
int main()
{
	int Size=5;
	char Array[Size]={'\0'};
	cout<<"Array Values: ";
	for(int i=0;i<Size;i++)
	{
	cin>>Array[i];
	}
	if(IsPalindrome(Array,Size))
	{
	cout<<"data is palindrome"<<endl;
	}
	else
	{
	cout<<"data is not palindrome"<<endl;
	}
	return 0;
}
int LenghtOfArray(char Array[ ],int Size)
{
	for(int i=0;i<Size;i++)
	{
	   if(Array[i]=='\0')
	   {
	     return i;
	   }
	}
}
bool IsPalindrome(char Array[ ],int Size)
{
	int Length=LenghtOfArray(Array,Size);
	for(int i=0;i<Length/2;i++)
	{
	if(Array[i]!=Array[Length-i-1])
	{
	return false;
	}
	}
	return true;
}
```

Link

Description

In this problem we have to check that whether the data enter is palindrome or not in other words we have to check that the word enter **is symmetric** or not we use the same **is symmetric** code in it. We firstly as you see in the main to take input as we do in the **integer Array**. Then we find out the **size** of **array** using **for loop** we initialize the **char array** as **\0** and we use null to get the size then we check the **Palindrome** using the size then gave the output on the screen.

Output

```
Array Values: madam
data is palindrome
```

Type 2 Using only cin

You can also cin a complete word using **cin>>** without loop but using it you can not enter a sentence.

Example

You have to check whether the word enter by the user are equal or not

Code

```
#include <iostream>
using namespace std;
int LenghtOfArray(char Array[ ],int Size);//It will Return the length of Array
bool IsEqual(char A[ ],char B[ ],int Size);//It will check if arrays are equal
int main()
{
        int Size=10;
        char A[Size]={'\0'};
        char B[Size]={'\0'};
        cout<<"enter Words: ";
        cin>>A>>B;
        if(IsEqual(A,B,Size))
        {
        cout<<"Words are equal"<<endl;
        }
        else
        {
        cout<<"Words are not equal"<<endl;
        }
        return 0;
}
int LenghtOfArray(char Array[ ],int Size)
{
        for(int i=0;i<Size;i++)
        {
          if(Array[i]=='\0')
         {
            return i;
         }
        }
}
bool IsEqual(char A[ ],char B[ ],int Size)
{
        int Length1=LenghtOfArray(A,Size);
        int Length2=LenghtOfArray(A,Size);
        if(Length1==Length2)
        {
          for(int i=0;i<Length1;i++)
          {
            if(A[i]!=B[i])
            {
                return false;
            }
          }
            return true;
        }
        return false;
```

```
48  }
```

Link

Description

In this we see the 2nd way to input in which you see we use a **cin** without a **loop** this input is used to enter the single word. In it we have to check that the enter both words are equal or not we start by first checking that whether there length is same or not if same **return true** else **return false** then if **true** we use a **for loop** to check whether the all values in the array are same or not if same then **true else false** and at end we gave **output** on the screen.

Output

```
enter Words: Zuraiz
Zuraiz
Words are equal
```

Type3 entering data in the program using single quotation marks

Code

```
1   #include <iostream>
2   using namespace std;
3   int LenghtOfArray(char Array[ ],int Size);//It will return Length of the array
4   void ReverseCopy(char A[ ],char B[ ],int Size);//It will reverse copy A data in B
5   bool IsEqual(char A[ ],char B[ ],int Size);//It will tell if arrays are equal
6   int main()
7   {
8           int Size=100;
9           char A[Size]="This Is This\0";
10          char B[Size]={'\0'};
11          ReverseCopy(A,B,Size);
12          if(IsEqual(A,B,Size))
13          {
14          cout<<"Data Entered Is Palindrome"<<endl;
15          }
16     else
17          {
18          cout<<"Data Entered Is not Palindrome"<<endl;
19          }
20          return 0;
21  }
22  int LenghtOfArray(char Array[ ],int Size)
23  {
24          for(int i=0;i<Size;i++)
25          {
26          if(Array[i]=='\0')
27          {
28          return i;
29          }
30          }
31  }
32  void ReverseCopy(char A[ ],char B[ ],int Size)
33  {
34          int Length=LenghtOfArray(A,Size);
35          int Re=Length;
36          int End=0;
37          int Ce=Re-1;
38          while(End!=Length)
39          {
40             for(Ce;A[Ce]!=' '&&Ce>=0;Ce--)
41             {
42             }
43             Ce++;
44             for(int C=Ce;A[C]!=' '&&C<Length;C++)
45             {
```

```
            if(End>=Length)
            {
            break;
            }
            B[End]=A[C];
            End++;
            }
            if(End>=Length)
          {
          break;
          }
            B[End]=' ';
            End++;
           Ce=Ce-2;
          }
}
bool IsEqual(char A[ ],char B[ ],int Size)
{
          int Length1=LenghtOfArray(A,Size);
          int Length2=LenghtOfArray(A,Size);
          if(Length1==Length2)
          {
             for(int i=0;i<Length1;i++)
             {
                if(A[i]!=B[i])
                {
                  return false;
                }
             }
             return true;
          }
          return false;
}
```

Link

Description

In this program we have used third technique to enter the data in it we hard code it as using single quotation we put our array equal to the data written in single quotation then after the data is entered we start building our program in which we use the logic that we first copy the sentence **reverse** in the other **array** and then only check if the both arrays are equal or not the **reverse copy example** is as follows enter data is equal to the **Hello world** and the reversed copy in B will be **world Hello**. For it we use a function Reverse copy in which firstly we use a **for loop** to move to our word start position then we copy it in other array start using another **for loop** and in this way we copy whole sentence using same technique in that **Array** after copying we check if both the arrays are **equal** or not and gave **output** on screen according to it.

Output

```
Data Entered Is Palindrome
```

8.5 File-streaming (Reading and Writing from file)

Till now we are getting input from user using console screen i.e. user enter from keyboard. There is another way to get input and store output in C++, which is called filing. In this method almost everything is same. The only change is that you have to include a C++ library **#include<fstream>** (file streaming) at start of program where you include other libraries. While taking input from console or displaying something on console we use **cin** and **cout** respectively, in file streaming we have to declare some variables using **ifstream** and **ofstream** data type which plays role like **cin** and **cout** in our program but this is used for filing.

Data Type	Description
ifstream	It is called input file stream. It is used to read data from file.
ofstream	It is called output file stream. It is used to create file and write data on file.

operator	fstream variables
<<	fout
>>	fin

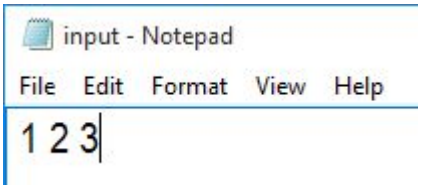
Getting input from file

If we want to get input from file, then we have to declare any variable like **fin** of data type **ifstream**. **ifstream** is similar to any other data type in C++ like **int**. It is only used to tell compiler that we want to read from file. There is a complete syntax on which we can read from file which is described in below example.

Code

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
        ifstream fin("input.txt");
        if( ! fin)
        {
            cout <<"file not open";
            return 0;
        }
        else
        {
            int a,b,c;
            fin >>a>>b>>c;
            cout<<"inputs from file are: \n";
            cout<<"a= "<<a<<" b= "<<b<<" c= "<<c<<endl;
            return 0;
        }
}
```

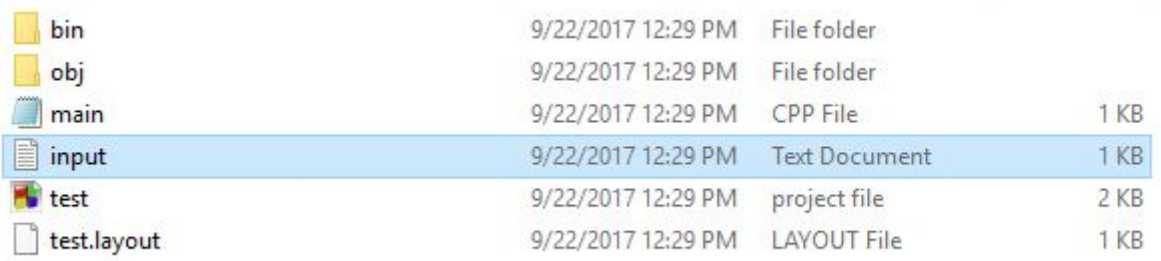
Input



Output

```
inputs from file are:
a= 1 b= 2 c= 3
```

Program folder



Link:

Description

In above code from line 1 to 3 we have defined some c++ libraries. From line 4 **int main( )** starts. At line 6 we have declared an **ifstream** variable **fin** which is used to get input from a file. At line 7 we have used an **if( )** check which become true if file is not open and then program terminates by displaying a message at console that file is not open. At line 12 we use **else( )** condition which get input from file and display that input on console.

Writing data to file

If we want to write data to file, then we have to declare any variable like **fout** of data type **ofstream**. **ofstream** is similar to any other data type in C++ like **int**. It is only used to tell compiler that we want to write data to file. There is a complete syntax on which we can write data to file which is described in below example.

Remember

If you want to store data in file and file is not available, then ofstream will make a file for you, but this facility is only available for ofstream, if you want to read data using ifstream then it is must that file is already available.

Code

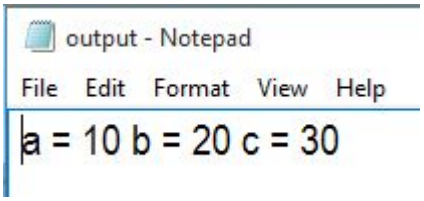
```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  int main()
5  {
6          int a, b, c;
7          cout << "enter any three numbers(space separated): ";
8          cin >> a >> b >> c;
9          ofstream fout("output.txt");
10         fout << "a = " << a << " b = " << b << " c = " << c << endl;
```

```
11          cout << "data is saved in a file." << endl;
12          return 0;
13  }
```

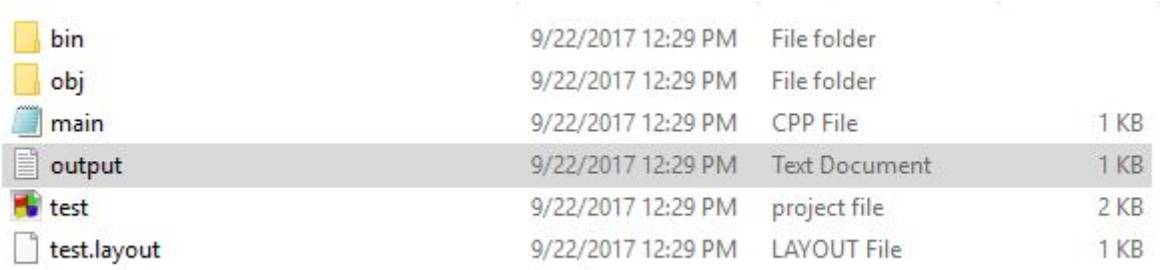
Output

```
enter any three numbers(space separated): 10 20 30
data is saved in a file.
```

Output file



Program folder



Link

Description

In above code from line 1 to 3 we have declared some c++ libraries. From line 4 **int main( )** starts. At line 6 we have declared three integers **a, b, c**. At line we display a message on console and at line 8 we get input from user in these declared variables. At line 9 we have declared an **ofstream** variable **fout** which is used to save data in a file. From line 10 and 11 we save a message and these variables in a file using **fout** variable. At line 12 we uses **return 0** statement to terminate this program.

8.5 Project: Electronic Voting

Election happened in a country with 8 parties fighting inside the election. Make a program to check who won the General Election. Read the data from **Votes.txt** file.

Code

```
1   #include<iostream>
2   #include<fstream>
3   using namespace std;
4   void read(ifstream & fin, int voters, int vote_cast[ ]);
5   void computeResult(int parties[ ], int vote_cast[ ], int voters);
6   int winnerCheck(int parties[ ]);
7   void displayResult(int parties[ ], char party_name[ ][5]);
8   void resultPlot(int parties[ ], char party_name[ ][5]);
9   int main()
10  {
11          ifstream fin("votes.txt");
12          int voters;
13          fin >> voters;
14          int vote_cast[voters];
15          int parties[9] = {0};
16          char party_name[9][5] = {"\0", "PPP", "PMLN", "PTI", "ANP", "JIP", "JUI", "MQM", "PAT"};
17          read(fin, voters, vote_cast);
```

```
18          computeResult(parties, vote_cast, voters);
19          displayResult(parties, party_name);
20          resultPlot(parties, party_name);
21          return 0;
22  }
23  void read(ifstream & fin, int voters, int vote_cast[ ])
24  {
25          for(int a=0; a<voters; a++)
26          {
27          fin >> vote_cast[a];
28          }
29  }
30  void computeResult(int parties[ ], int vote_cast[ ], int voters)
31  {
32          for(int b=0; b<voters; b++)
33          {
34          int currentVote = vote_cast[b];
35          parties[currentVote] = parties[currentVote] + 1;
36          }
37  }
38  int winnerCheck(int parties[ ])
39  {
40          int max = parties[0];
41          int index = 0;
42          for(int a=1; a<9; a++)
43          {
44          if(parties[a] > max)
45          {
46          max = parties[a];
47          index = a;
48             }
49          }
50          return index;
51  }
52  void displayResult(int parties[ ], char party_name[ ][5])
53  {
54          cout << "Party Name" << "\t" << "Parties" << "\t\t" << "Votes" << endl;
55          for(int p=1; p<=8; p++)
56          {
57          cout << party_name[p] << "\t\t" << p << "\t\t" << parties[p] << endl;
58          }
59          cout << endl;
60  }
61  void resultPlot(int parties[ ], char party_name[ ][5])
62  {
63          char sym = -37;
64          for(int plot=1; plot<=8; plot++)
65          {
66          cout << party_name[plot] << "  ";
67          for(int p=0; p<parties[plot]; p++)
68          {
69          cout << sym;
70          }
71          cout << endl << endl;
72          }
73          cout << endl << party_name[winnerCheck(parties)] << " party won the election." << endl;
74  }
```

Output

```
Party Name      Parties         Votes
PPP             1               12
PML             2               3
PTI             3               2
ANP             4               2
JIP             5               5
JUI             6               9
MQM             7               5
PAT             8               2

PPP ██████████

PML ███

PTI ██

ANP ██

JIP ████

JUI ███████

MQM ████

PAT ██


PPP party won the election.
```

Description

Practice Exercise

1. In the 20×20 grid below (read from file), four numbers along a diagonal line have been marked in red.

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

The product of these numbers is 26 × 63 × 78 × 14 = 1788696.

What is the greatest product of four adjacent numbers in the same direction (up, down, left, right, or diagonally) in the 20×20 grid?