DASF004: Basic and Practice in Programming

- Lab 11: File Input/Output

In this lab ...

- Be familiar with various operations and functions in reading and writing file input/output

- What you need to submit in this lab (Lab #11):
  - » Lab Exercise #11 by Wednesday 11:59 pm
  - » Assignment #11 by Tuesday 11:59 pm

Procedure for read from or writing to a file

Four Step Procedure

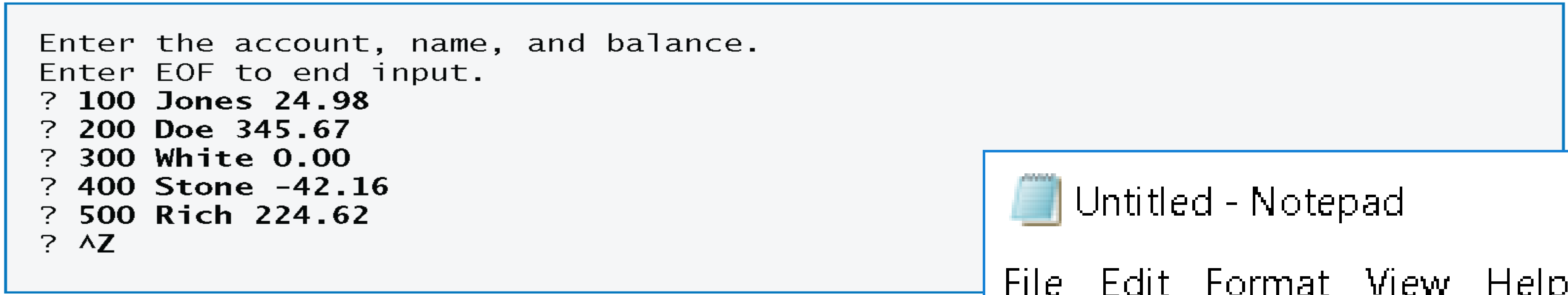
- 1. Declare a file pointer
  - » `FILE *fPtr;`
- 2.Call fopen(), specify the filename, and the access permission
  - » `fPtr = fopen("[filename]","[permission]");`
- 3. Do the read/write operation
  - » `fprintf()`
  - » `fscanf()`
  - » `fread()`
  - » `fwrite()`
  - » `fseek()`
- 4.Close the file
  - » `fclose(fPtr);`

File Access

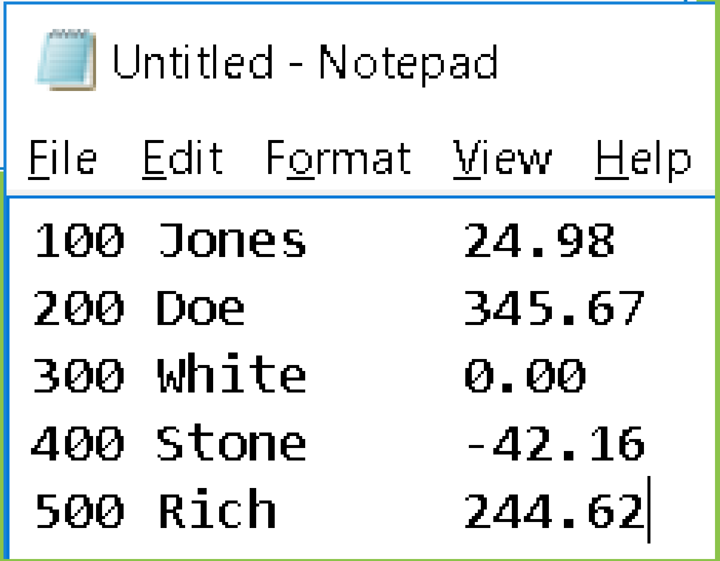
There are two kinds of file access

1. Sequential Access
   - You have to read and write the file starting from the beginning, reading and writing byte by byte from the start of the file
2. Random Access
   - You can read and write any byte in the file
   - You control a file pointer to select the byte you want to read/write using `fseek()`

Examples



```
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```



```
100 Jones        24.98
200 Doe          345.67
300 White        0.00
400 Stone        -42.16
500 Rich         244.62
```

```c
// Fig. 11.2: fig11_02.c
// Creating a sequential file
#include <stdio.h>

int main( void )
{
   unsigned int account; // account number
   char name[ 30 ]; // account name
   double balance; // account balance

   FILE *cfPtr; // cfPtr = clients.dat file pointer

   // fopen opens file. Exit program if unable to create file
   if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
      puts( "File could not be opened" );
   } // end if
   else {
      puts( "Enter the account, name, and balance." );
      puts( "Enter EOF to end input." );
      printf( "%s", "? " );
      scanf( "%d%29s%lf", &account, name, &balance );

      // write account, name and balance into file with fprintf
      while ( !feof( stdin ) ) {
         fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
         printf( "%s", "? " );
         scanf( "%d%29s%lf", &account, name, &balance );
      } // end while

      fclose( cfPtr ); // fclose closes file
   } // end else
} // end main
```

Mode	Description
r	Open an existing file for reading.
w	Create a file for writing. If the file already exists, discard the current contents.
a	Append: open or create a file for writing at the end of the file.
r+	Open an existing file for update (reading and writing).
w+	Create a file for update. If the file already exists, discard the current contents.
a+	Append: open or create a file for update; writing is done at the end of the file.
rb	Open an existing file for reading in binary mode.
wb	Create a file for writing in binary mode. If the file already exists, discard the current contents.
ab	Append: open or create a file for writing at the end of the file in binary mode.
rb+	Open an existing file for update (reading and writing) in binary mode.
wb+	Create a file for update in binary mode. If the file already exists, discard the current contents.
ab+	Append: open or create a file for update in binary mode; writing is done at the end of the file.

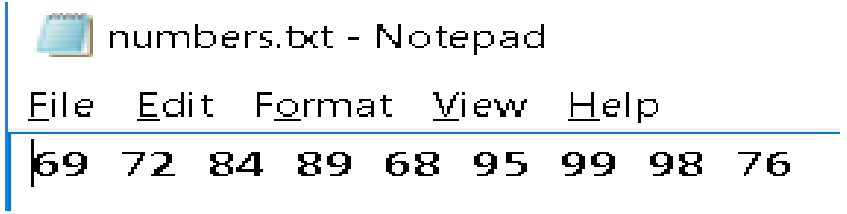
`rewind()` function

- The statement
    - `rewind( fPtr );`

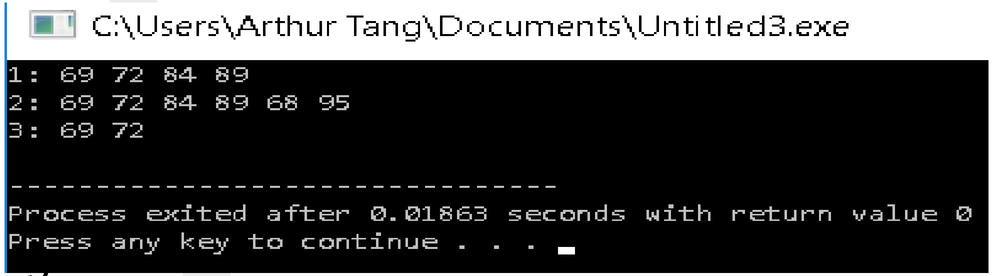
  causes a program’s file position pointer—which indicates the number of the next byte in the file to be read or written—to be repositioned to the *beginning* of the file (i.e., byte 0) pointed to by `fPtr`.

rewind()

```c
#include <stdio.h>

int main(void)
{ FILE *fPtr;
  int x[9];
  fPtr = fopen("numbers.txt","r");
  fscanf(fPtr,"%d %d %d %d",&x[0],&x[1],&x[2],&x[3]);
  printf("1: %d %d %d %d\n",x[0],x[1],x[2],x[3]);
  rewind(fPtr);

  fscanf(fPtr,"%d %d %d %d %d %d",&x[0],&x[1],&x[2],&x[3],&x[4],&x[5]);
  printf("2: %d %d %d %d %d %d\n",x[0],x[1],x[2],x[3],x[4],x[5]);
  rewind(fPtr);

  fscanf(fPtr,"%d %d",&x[0],&x[1]);
  printf("3: %d %d\n",x[0],x[1]);

  fclose(fPtr);
  return 0;
}
```



```
69 72 84 89 68 95 99 98 76
```



```
1: 69 72 84 89
2: 69 72 84 89 68 95
3: 69 72

--------------------------------
Process exited after 0.01863 seconds with return value 0
Press any key to continue . . .
```

Binary Records

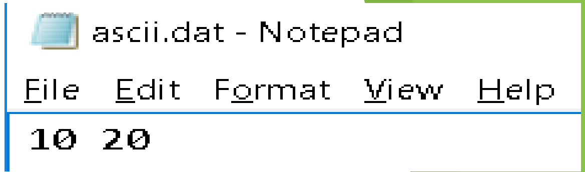
- ASCII Records
  - Everything is represented as text
  - The number “10” is represented by the ASCII character “1”,”0”
    - 1: 00100001; 0: 00100000
- Binary Records
  - Records are in 1 and 0 only (i.e. not in text form)
  - Integers are represented in their binary value
    - Integer 10 is represented by 00000000 00000000 00000000 00001010
  - They cannot be viewed in notepad

Binary Records Example

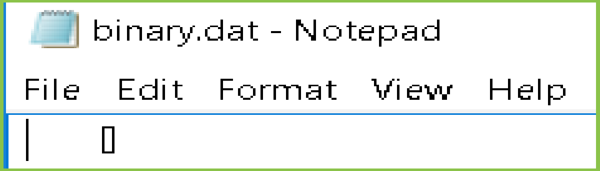
```
#include <stdio.h>
int main(void)
{ int x[2] = {10,20};
  FILE *f1Ptr = fopen("ascii.dat","w");
  FILE *f2Ptr = fopen("binary.dat","wb");

  fprintf(f1Ptr,"%d %d",x[0],x[1]);
  fwrite(&x,sizeof(int),2,f2Ptr);



  fclose(f1Ptr);
  fclose(f2Ptr);
  return 0;
}
```

Write the string "10 20"
(the bit stream: 00100001 00100000 00100000 00100010 00100000)

Write the binary of 10 and 20
(the bit stream: 00000000 00000000 00000000 00001010 00000000 00000000 00000000 00010100)



```
10 20
```

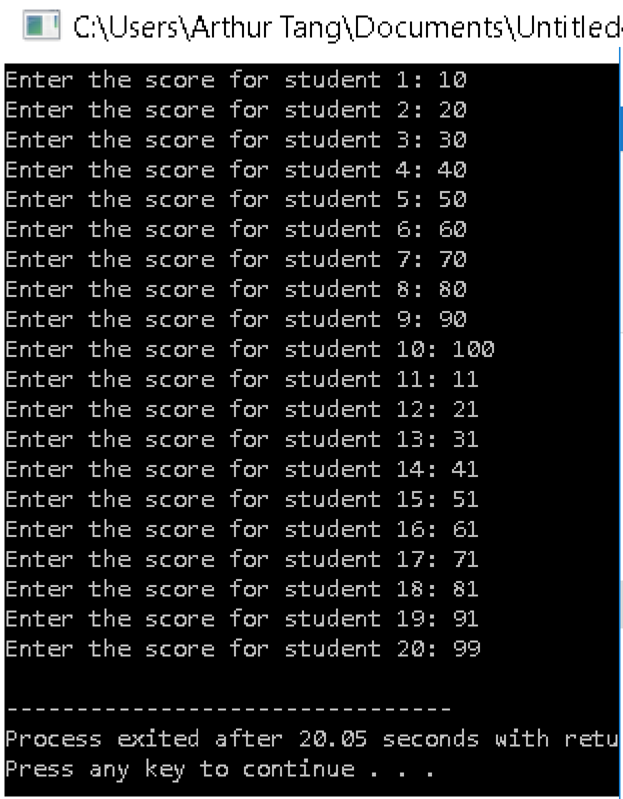


```
|    ▯
```

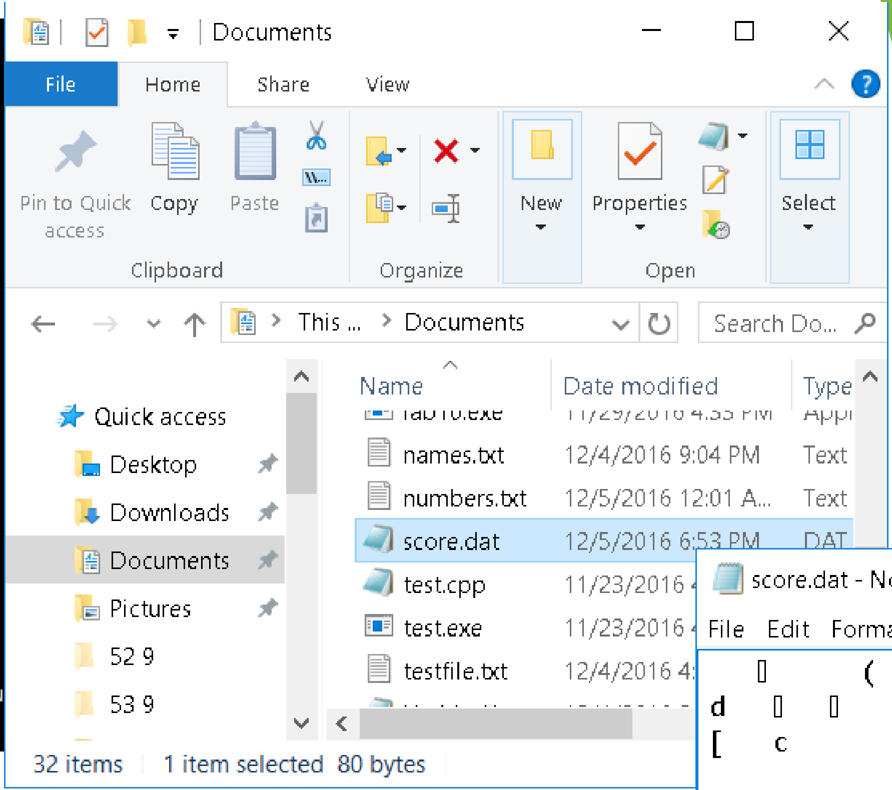
Lab Exercise #11:

- Write a program ask for the user to enter the score (integer) for 20 students (student 1 to student 20).
- Your program will write these 20 integers in binary form into a file name “score.dat”

Sample Output

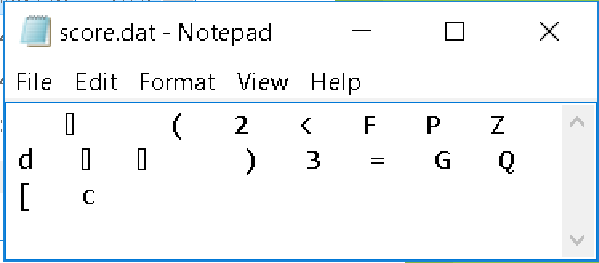

C:\Users\Arthur Tang\Documents\Untitled4.exe

```
Enter the score for student 1: 10
Enter the score for student 2: 20
Enter the score for student 3: 30
Enter the score for student 4: 40
Enter the score for student 5: 50
Enter the score for student 6: 60
Enter the score for student 7: 70
Enter the score for student 8: 80
Enter the score for student 9: 90
Enter the score for student 10: 100
Enter the score for student 11: 11
Enter the score for student 12: 21
Enter the score for student 13: 31
Enter the score for student 14: 41
Enter the score for student 15: 51
Enter the score for student 16: 61
Enter the score for student 17: 71
Enter the score for student 18: 81
Enter the score for student 19: 91
Enter the score for student 20: 99

--------------------------------
Process exited after 20.05 seconds with retu
Press any key to continue . . .
```



Name	Date modified	Type
lab10.exe	11/29/2016 4:35 PM	Appl
names.txt	12/4/2016 9:04 PM	Text
numbers.txt	12/5/2016 12:01 A...	Text
score.dat	12/5/2016 6:53 PM	DAT
test.cpp	11/23/2016	
test.exe	11/23/2016	
testfile.txt	12/4/2016 4:	

32 items 1 item selected 80 bytes



```
 ▯      (   2   <   F   P   Z
d ▯  ▯     )   3   =   G   Q
[  c
```

The `fread()` function

```
fread(&myrec,sizeof(int),6,fPtr);
```

- Size of each block is size of int (i.e. 4 byte)
- Read 6 blocks (i.e. 24 byte)
- Read from the **current location** of the file pointer pointed to by `fPtr`
- The block read is stored to the memory address &myrec.

The `fwrite()` function

```
fwrite(&myrec,sizeof(int),6,fPtr);
```

Size of each block is size of int (i.e. 4 byte)

Write 6 blocks (i.e. 24 byte)

Write to the **current location** of the file pointer pointed to by `fPtr`.

The block written is from the memory address &myrec.

What is the current location???

Initially, the current location is the first byte of the file

The location pointer moves everytime you do a `fread()` or `fwrite()`

- The pointer is moved to the byte after the last byte you read/write

For example:

```
fwrite(&myrec,sizeof(int),6,fPtr);
```

The pointer is at the beginning of the file first.

`fwrite()` writes 6 block of sizeof(int) at the pointer (24 byte)

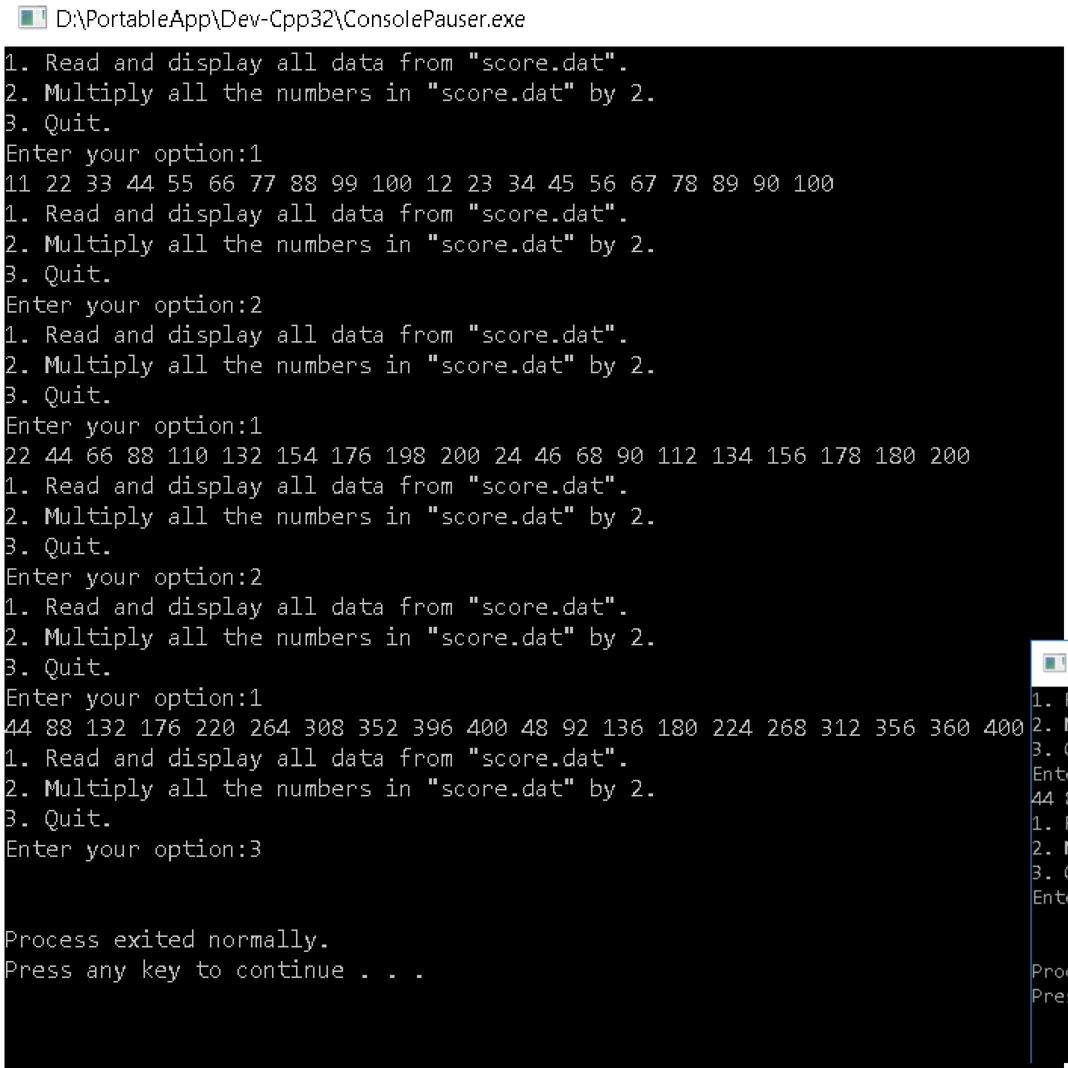
After the write, pointer's current location is at 25th byte of the file

Lab Assignment #11:

- Your program will read the file “score.dat” you generated in Lab Exercise #11. Note that the file is in **binary form**.

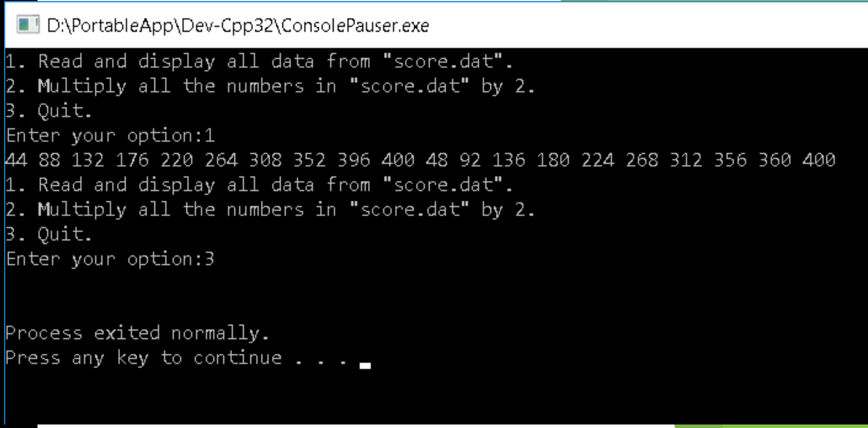
- Your program will then prompt the user for 3 options:
  1. Display all the numbers in the file “score.dat”.
  2. Multiply all the numbers in the file by 2.
  3. Quit
- Option 1 will read and display all numbers saved in “score.dat”.
- If the user enter option 2, your program will read each number from “score.dat”, multiply each number by 2, and **write the number back to “score.dat”**
- Note that all modifications are written to the score.dat file. So if you restart your program, all the changes in the previous run will remind in the file.

Sample output



D:\PortableApp\Dev-Cpp32\ConsolePauser.exe

```
1. Read and display all data from "score.dat".
2. Multiply all the numbers in "score.dat" by 2.
3. Quit.
Enter your option:1
11 22 33 44 55 66 77 88 99 100 12 23 34 45 56 67 78 89 90 100
1. Read and display all data from "score.dat".
2. Multiply all the numbers in "score.dat" by 2.
3. Quit.
Enter your option:2
1. Read and display all data from "score.dat".
2. Multiply all the numbers in "score.dat" by 2.
3. Quit.
Enter your option:1
22 44 66 88 110 132 154 176 198 200 24 46 68 90 112 134 156 178 180 200
1. Read and display all data from "score.dat".
2. Multiply all the numbers in "score.dat" by 2.
3. Quit.
Enter your option:2
1. Read and display all data from "score.dat".
2. Multiply all the numbers in "score.dat" by 2.
3. Quit.
Enter your option:1
44 88 132 176 220 264 308 352 396 400 48 92 136 180 224 268 312 356 360 400
1. Read and display all data from "score.dat".
2. Multiply all the numbers in "score.dat" by 2.
3. Quit.
Enter your option:3


Process exited normally.
Press any key to continue . . .
```



D:\PortableApp\Dev-Cpp32\ConsolePauser.exe

```
1. Read and display all data from "score.dat".
2. Multiply all the numbers in "score.dat" by 2.
3. Quit.
Enter your option:1
44 88 132 176 220 264 308 352 396 400 48 92 136 180 224 268 312 356 360 400
1. Read and display all data from "score.dat".
2. Multiply all the numbers in "score.dat" by 2.
3. Quit.
Enter your option:3


Process exited normally.
Press any key to continue . . .
```

Lab Assignment #11

Submit your source code on iCampus before Tuesday 11:59 pm