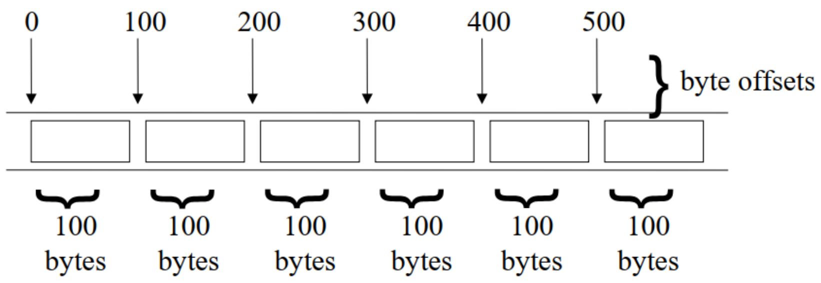
Section #2 – File Organization

Indexed- Files

1) Random Files:

1. Access individual records without searching through other records.
2. Instant access to records in a file.
3. Data can be inserted without destroying other data.
4. Data previously stored can be updated or deleted without overwriting.
5. Random access files are implemented using fixed-length records.
6. Sequential files do not have fixed-length records.

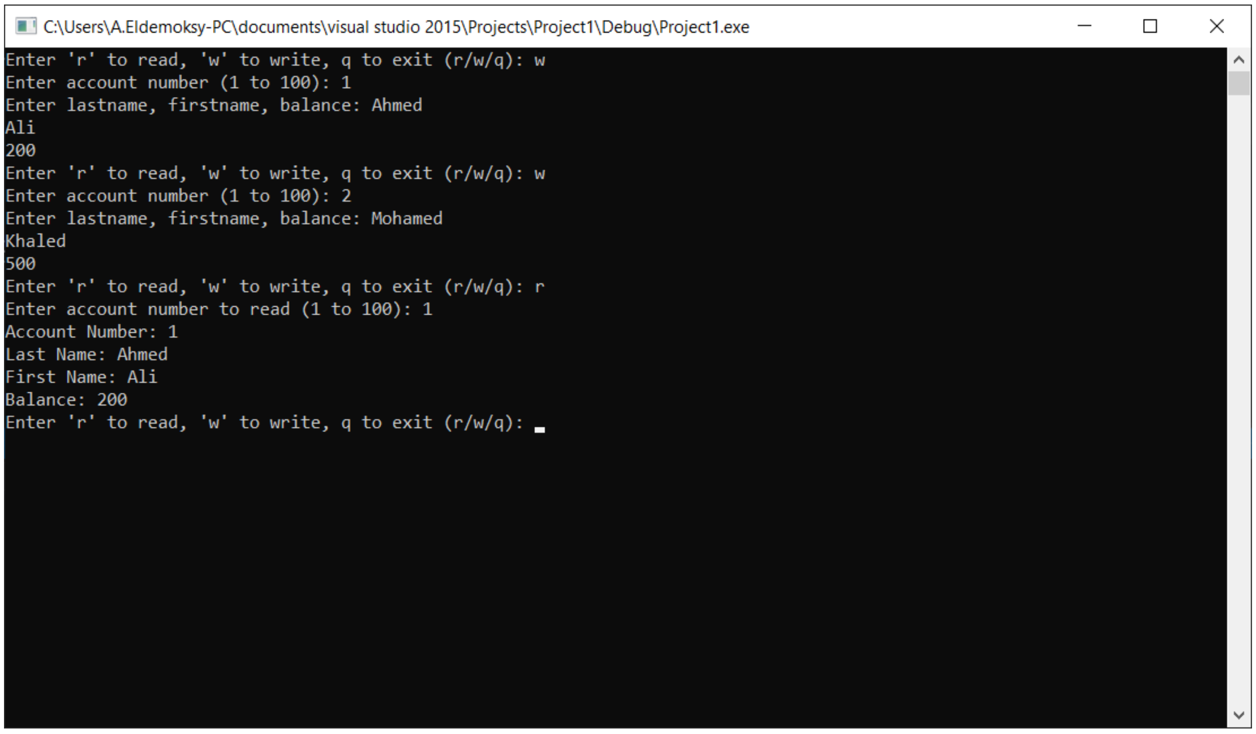


2) A C++ application for storing client data that reads and writes data at a specified location using seekg and seekf.

```
#include <iostream>
#include <fstream>
#include <cstring>

using namespace std;

// clientData structure definition
struct ClientData {
      int acctNum;             // account number
      char lastName[15];       // account last name
      char firstName[10];      // account first name
      double balance;          // account balance
}; // end structure ClientData
```

```
void main() {
	ClientData client; // clientData object

	// fstream object for reading and writing to a file
	fstream file("d:/credit.dat", ios::in | ios::out | ios::binary);

	// open file for reading and writing;
	if (!file) {
		cerr << "File could not be opened." << endl;
		return;
	}
	else {
		// require user to specify action: read, write
		char action = 'a';
		while (action != 'q')
		{
		cout << "Enter 'r' to read, 'w' to write, q to exit (r/w/q): ";
		cin >> action;

		if (action == 'w') { // Write to file
			// require user to specify account number
			cout << "Enter account number (1 to 100): ";
			cin >> client.acctNum;

			// user enters information until account number is 0
				// user enters last name, first name, and balance
			cout << "Enter lastname, firstname, balance: ";
			cin >> client.lastName >> client.firstName
			    >> client.balance;

			// seek position in file of user-specified record
			file.seekp((client.acctNum - 1) * sizeof(ClientData),
			                                  ios::beg);

			// write user-specified information in file
			file.write(reinterpret_cast<const char*>(&client),
			                               sizeof(ClientData));
		}
		else if (action == 'r') { // Read from file
		// require user to specify account number to read
			cout << "Enter account number to read (1 to 100): ";
			int accountToRead;
			cin >> accountToRead;
```

```
                        // seek position in file of user-specified record
                        file.seekg((accountToRead - 1) * sizeof(ClientData),
                                                          ios::beg);

                        // read data from file
                        file.read(reinterpret_cast<char*>(&client),
                                                          sizeof(ClientData));

                        // display the data
                        cout << "Account Number: " << client.acctNum << endl;
                        cout << "Last Name: " << client.lastName << endl;
                        cout << "First Name: " << client.firstName << endl;
                        cout << "Balance: " << client.balance << endl;
                }
                else if (action == 'q') {
                        exit(0);
                }
                else
                        cout << "invalid" << endl;
                }
                file.close(); // close the file
        }
        system("pause");
}
```


C:\Users\A.Eldemoksy-PC\documents\visual studio 2015\Projects\Project1\Debug\Project1.exe

```
Enter 'r' to read, 'w' to write, q to exit (r/w/q): w
Enter account number (1 to 100): 1
Enter lastname, firstname, balance: Ahmed
Ali
200
Enter 'r' to read, 'w' to write, q to exit (r/w/q): w
Enter account number (1 to 100): 2
Enter lastname, firstname, balance: Mohamed
Khaled
500
Enter 'r' to read, 'w' to write, q to exit (r/w/q): r
Enter account number to read (1 to 100): 1
Account Number: 1
Last Name: Ahmed
First Name: Ali
Balance: 200
Enter 'r' to read, 'w' to write, q to exit (r/w/q): _
```