

Consider the problem of organizing and manipulating the following data:

Origin	Destination	Miles	Time
Blacksburg, VA	Knoxville, TN	244	3:25
Knoxville, TN	Nashville, TN	178	2:35
Nashville, TN	Memphis, TN	210	3:17
Memphis, TN	Little Rock, AR	137	2:05
Little Rock, AR	Texarkana, TX	141	2:10

Suppose that we need to calculate some values for each trip, such as the average speed. Also suppose that we need to be able to look up all trips with a given origin and report the data for those trips. How can we accomplish this?

Clearly we must read in all of the trip data and store it in some way; we cannot just process this data line by line because reading data from a file on disk is too slow for a useful application.

However, we cannot store all the data in a single array since an array can hold data of only one type, and we must deal with strings and integers and decimal numbers.

The solution does involve using arrays... note that the data is naturally organized as a table, where each row represents a particular trip and all the values in each column are of the same type.

We may organize a table of related data of differing types by using a collection of arrays, thinking of each array as representing a column of the table.

The values from each row of the table would be stored across the various arrays, but at the same index value in each.

By declaring appropriate arrays, we may read and store the given data as shown below, and then calculate the speed for each trip and store those values:

```
const int MAXTRIPS = 500;
string Origin[MAXTRIPS];
string Destination[MAXTRIPS];
int Miles[MAXTRIPS];
int Minutes[MAXTRIPS];
double MPH[MAXTRIPS];
```

	Origin	Destination	Miles	Minutes	MPH
0	Blacksburg, VA	Knoxville, TN	244	205	??
1	Knoxville, TN	Nashville, TN	178	155	??
2	Nashville, TN	Memphis, TN	210	197	??
3
4	Little Rock, AR	Texarkana, AR	141	160	??

The only special issue when using parallel arrays is to be careful to always access each array at the same index when storing or retrieving values, in order to reference corresponding data locations:

```
void printTrips(ofstream& Out, const int numTrips, const string Origin[],
               const string Destination[], const int Miles[],
               const int Minutes[], const double MPH[]) {

    for (int Idx = 0; Idx < numTrips; Idx++) {

        Out << left << setw(MAXNAMELENGTH + 1) << Origin[Idx]
            << setw(MAXNAMELENGTH + 1) << Destination[Idx]
            << right << setw(10) << Miles[Idx]
            << setw(10) << (Minutes[Idx] / MINSPEERHOUR) << ':'
            << setfill('0') << setw(2) << (Minutes[Idx] % MINSPEERHOUR)
            << setfill(' ') << setw(10) << setprecision(1)
            << MPH[Idx]
            << endl;

    }
}
```

```
void readTrips(ifstream& In, const int maxTrips, int& numTrips,
              string Origin[], string Destination[], int Miles[],
              int Minutes[]) {

    int Line = 0;                // count of lines read
    string Orig, Dest;          // local temp storage for values read in
    int Mi, Min;

    readOneLine(In, Orig, Dest, Mi, Min);    // priming read

    while ( In && Line < maxTrips ) {

        Origin[Line] = Orig;          // put new data into arrays
        Destination[Line] = Dest;
        Miles[Line] = Mi;
        Minutes[Line] = Min;

        ++Line;                      // count latest line read

        readOneLine(In, Orig, Dest, Mi, Min); // read next line
    }
    numTrips = Line;                // return # of lines read
}
```