Arrays

Lab 12: Maze Routing with Two Dimensional Arrays

Rung-Bin Lin

International Bachelor Program in Informatics
Yuan Ze University

Dec. 12, 2023

Arrays

- One dimensional arrays vs. multi-dimensional arrays.
- How to declare an array?
- How to put data into arrays?
- How to pass arrays to a function?
- How to process data in arrays?

How to declare an array?

- The array size must be known upon declaration.

```
int anIntAry[10];
const int arraySize = 20;
string anStringAry[arraySize];
string aTwoDimStringAry[10][20];
const int numOfRows = 15;
const int numOfColumns = 25;
int aTwoDimIntAry[numOfRows][numOfColumns];
double aThreeDimDoubleAry[10][5][24];
```

How to initialize an array?

- Initialize it once during declaration

```
int anIntAry[5]  = {1,2,3,4,5};
string anStringAry[5] = {“1ST”, “2ND”, “3rd”}
int aTwoIntAry[4][3] ={ };
int antTwoIntAry[4][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
int antTwoIntAry[4][3] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

- Initialize it during execution

```
for(int i=0;i<4; i++)
      for(int j=0; j<3; j++)
           aTwoIntAry[i][j] = i+j;
```

	Column 0	Column 1	Column 2
Row 0	0	1	2
Row 1	1	2	3
Row 2	2	3	4
Row 3	3	4	5

Pass arrays to a function in a function call

- Pass by reference, so only the starting address of an array is passed.
- How to specify the starting address of an array?
  - **Use its name or the address of the first element**
    - **For one-dimensional array**
      &arrayOne[0] or arraryOne
    - **For two dimensional array**
      &arrayTwo[0][0] or arrayTwo
    - **For three dimensional array**
      &arrayThree[0][0][0] or arrayThree

Declare an array in a function's parameter list

- **Declare an array in a function prototype**

  The size of each dimension except the first dimension must be given.

```
const int dim1 = 10;
const int dim2 = 20;
const int ndim3 = 30;
void aFunc( int [ ], int x[ ], int y[ ][dim2], int [ ][dim2][dim3],
int);
```

Must be constant integers

- **Declare an array in the parameter list of a function body**

```
void aFunc(int a[ ], int b[ ], int c[ ][dim2], int d[ ][dim2][dim3],
int dim1)
{
...
}
```

Should be dim2 and dim3

Used for specifying the size of the first dimension

It is incorrect to declare a function shown below to pass arrays to the function:

```
void aFunc(int a[ dim1], int b[ dim1], int c[ dim1][dim2],
int d[dim1 ][dim2][dim3], int dim1)
```

Make calls to a function

```
const int dim1 = 40, dim2 = 20, dim3 = 30;
void aFunc( int [ ], int x[ ], int y[ ][dim2], int [ ][dim2][dim3], int);
int main(){
    int w[28], x[dim1], y[dim1]dim2], z[dim1]dim2][dim3];
    int dim = 28
    aFunc( w, &x[0], &y[0][0], z, dim);
        …
}
void aFunc(int a[ ], int b[ ], int c[ ][dim2], int d[ ][dim2][dim3], int dima)
{
   for(int x=0; x<dima; x++)
        a[x] = x;
   for(int d1=0; d1<dima; d1++){
      b[d1] = d1+2;
      for(int d2=0; d2<dim2; d2++){
         c[d1][d2] = d1 * d2 +13;
         for(int d3=0;d3<dim3; d3++)
            d[d1][d2][d3] = d1+d2 * d3 + 23;
      }
   }
}
```

Lab 12: Maze Routing

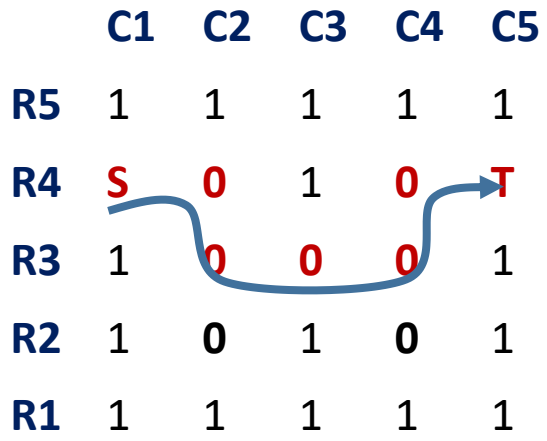
- **Given a maze of M rows and N columns of grids. Each entry in the maze contains either a 0, 1, S or T, where 1 denotes an obstacle, 0 denotes a free grid, S is a source grid, and T is a target grid. Assume there is only one source and one target grid on the boundary of the maze. *However, a source (target) grid is never located at a corner.* Write a program using the algorithm given below to find a path from S to T. Print the total distance travelled and the coordinates (row, column) of the start and target. If no target is found, treat the source as the target.**
- **You should used the following algorithm to solve the problem.**

  S1: Set current grid to S and depart from the source (S).

  S2: Following the traveling direction, if the grid on the left of the current grid is free, then turn left and advance to it. Otherwise, if the grid just ahead of the current grid is free, go straight to it. Otherwise, if the grid on the right of the current grid is free, then turn right and advance to it. Otherwise, reverse the direction and proceed backward.

  S3: Repeat S2 until hit the target T or return back to the source S.

- **For example, given the maze on the right, your program should find the path marked with red symbols. The start's and target's coordinates are (4,1) and (4, 5). The curve shows the grids being walked. The distance travelled is 6, which is calculated based on this curve.**



Input Format

- The first line gives the number of test cases. The first line in each test case gives M and N, i.e., the size of a maze. Then, it is followed by M rows of data. Each row taking a line has N symbols of 0, 1, S or T separated by blank spaces. *The first row will be R1, the second row will be R2, and the k-th row will be Rk*. Assume that RM is the top most row, as shown in the example of the previous slide. The column is numbered from left to right.

- Input Example

```
6
5 5
1 1 1 1 1
1 0 1 0 1
1 0 0 0 1
S 0 1 0 T
1 1 1 1 1
7 7
1 1 1 S 1 1 1
1 0 0 0 0 0 1
1 0 0 1 1 1 T
1 0 0 0 0 0 1
1 0 1 1 0 1 1
1 0 0 1 0 1 1
1 1 1 1 1 1 1
```

```
6 8
1 1 T 1 1 1 1 1
1 0 0 1 1 0 1 1
1 0 0 0 0 0 0 1
1 0 1 0 1 0 0 S
1 0 0 0 1 0 1 1
1 1 1 1 1 1 1 1
10 10
1 1 1 1 1 1 1 T 1 1
1 1 0 1 0 1 0 0 0 1
1 0 0 1 0 1 0 0 0 1
1 0 0 0 0 1 1 0 1 1
1 1 0 1 0 0 0 0 0 1
1 0 0 1 0 1 1 0 1 1
1 0 0 0 0 0 1 0 0 1
1 0 0 1 0 1 1 0 1 1
1 1 0 1 0 1 0 0 0 1
1 1 S 1 1 1 1 1 1 1
```

```
6 9
1 1 S 1 1 T 1 1 1
1 0 0 1 1 0 1 1 1
1 0 1 0 0 0 0 1 1
1 0 1 0 1 0 0 0 1
1 0 0 0 1 0 1 1 1
1 1 1 1 1 1 1 1 1
10 10
1 1 1 1 1 1 1 1 S 1
1 1 0 1 0 1 0 0 0 1
1 0 0 1 0 1 0 0 0 1
1 0 0 0 0 1 1 0 1 1
1 1 0 1 0 0 0 0 0 1
1 0 0 1 0 1 1 0 1 1
T 0 0 0 0 0 1 0 0 1
1 0 0 1 0 1 1 0 1 1
1 1 0 1 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1 1
```

Output Format

- The output of each test case takes a line starting with a '#'. See the example below for details.

```
6
5 5
1 1 1 1 1
1 0 1 0 1
1 0 0 0 1
S 0 1 0 T
1 1 1 1 1
# Source coordinates: (4, 1).  Target coordinates: (4, 5), Distance traveled: 6
7 7
1 1 1 S 1 1 1
1 0 0 0 0 0 1
1 0 0 1 1 1 T
1 0 0 0 0 0 1
1 0 1 1 0 1 1
1 0 0 1 0 1 1
1 1 1 1 1 1 1
# Source coordinates: (1, 4).  Target coordinates: (1, 4), Distance traveled: 30
6 8
1 1 T 1 1 1 1 1
1 0 0 1 1 0 1 1
1 0 0 0 0 0 0 1
1 0 1 0 1 0 0 S
1 0 0 0 1 0 1 1
1 1 1 1 1 1 1 1
# Source coordinates: (4, 8).  Target coordinates: (1, 3), Distance traveled: 10
10 10
1 1 1 1 1 1 1 T 1 1
1 1 0 1 0 1 0 0 0 1
1 0 0 1 0 1 0 0 0 1
1 0 0 0 0 1 1 0 1 1
1 1 0 1 0 0 0 0 0 1
1 0 0 1 0 1 1 0 1 1
1 0 0 0 0 0 1 0 0 1
1 0 0 1 0 1 1 0 1 1
1 1 0 1 0 1 0 0 0 1
1 1 S 1 1 1 1 1 1 1
# Source coordinates: (10, 3).  Target coordinates: (1, 8), Distance traveled: 38
6 9
1 1 S 1 1 T 1 1 1
1 0 0 1 1 0 1 1 1
1 0 1 0 0 0 0 1 1
1 0 1 0 1 0 0 0 1
1 0 0 0 1 0 1 1 1
1 1 1 1 1 1 1 1 1
# Source coordinates: (1, 3).  Target coordinates: (1, 6), Distance traveled: 21
10 10
1 1 1 1 1 1 1 1 S 1
1 1 0 1 0 1 0 0 0 1
1 0 0 1 0 1 0 0 0 1
1 0 0 0 0 1 1 0 1 1
1 1 0 1 0 0 0 0 0 1
1 0 0 1 0 1 1 0 1 1
T 0 0 0 0 0 1 0 0 1
1 0 0 1 0 1 1 0 1 1
1 1 0 1 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1 1
# Source coordinates: (1, 9).  Target coordinates: (7, 1), Distance traveled: 28
```

main() Function

```
int main()
{
   int numTest; // Number of test cases
   char maze[dim1][dim2]; // array for a maze map
   cin >> numTest;
   for(int i =0; i<numTest; i++){
      int column;  // number of columns
      int row;  // number of rows
      int numOfMove;  // distance traveled
      int startX;  // X coordinate of a source
      int startY;  // Y coordinate of a source
      int targetX;  // X coordinate of a target
      int targetY;  // Y coordinate of a target
      cin >> row >> column;
      readMaze(maze, row, column, startX, startY );  // a function to read in maze data and pass back the coordinates of the
      numOfMove = findPath(maze, row, column, startX, startY, targetX, targetY); // a function to find a path from S to T,
                                                                                 // pass back  the coordinates of the targ
                                                                                 // and return the distance traveled
      cout <<  "# Source coordinates: (" << startX+1 << ", " << startY+1 <<  ").   Target coordinates: ("
         << targetX+1<< ", " << targetY+1<< "), Distance traveled: " << numOfMove <<endl;
   }
   return 0;
}
```

Requirements

- **The main() function should not be modified.**
- **Need to implement two functions that will fulfill the two calls *readMaze(…)* and *findPath(…)* in the main function, respectively.**