CSCI 2132
Software Development

Lecture 16:

Multidimensional Arrays

Instructor: Vlado Keselj

Faculty of Computer Science

Dalhousie University

Previous Lecture

- Software testing
- Software debugging, gdb
- Arrays in C

Example: `binary.c`

- We will see an example in binary search
- Write program to:
  - Enter 10 numbers in increasing order
  - Enter a number to search
  - Report position in which it was found, or not found
- We will look at code with fill-in blanks

```
/* CSCI 2132 Sample "fill-in blanks" code: binary.c */
#include <stdio.h>

#define LEN 10

int main() {
  int array[LEN], lower, upper, middle, key, i;

  printf("Enter %d numbers in ascending order:\n",LEN);

  for (i = 0; i < LEN; i++)
    scanf("%d", _________ );

  printf("Enter the number to be searched for: ");
  scanf("%d", &key);
```

```
lower  = 0;
upper  = LEN - 1;
middle = (lower + upper) / 2;

while (array[middle] != key && lower < upper) {

  if      (lower+1 == upper)    lower = ___________ ;
  else if (key < array[middle]) upper = ___________ ;
  else                          lower = ___________ ;

  middle = (lower + upper) / 2;
}

if ( ____________ )
  printf("%d is the %d-th number you entered.\n",
         key, middle+1);
else
  printf("Not found.\n");
```

```
    return 0;
}
```

Multidimensional Arrays

- C allows multidimensional arrays; for example:

```
int m[5][9];
```

- defined a 2D array 5×9
- We can access elements from `m[0][0]` to `m[4][8]`
- Element in row 1, column 4 would be (fill): `m[  ][  ]`
- Stored in memory using row-major order
- Initialization example:

```
int t[3][3] = {{1, 0, 0},
               {0, 1, 0},
               {0, 0, 1}};
```

More about Multidimensional Array Initialization

- Inner braces can be omitted; e.g.:

```
int a[3][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
```

- or, e.g.:

```
int b[3][3] = { 0 };
```

Variable-Length Arrays (C99)

- C99 introduced variable-length arrays:
  - Arrays with non-constant length at compile time
- VLA Example:

```
int len, i;
printf("Enter the number of integers: ");
scanf("%d", &len);

int array[len];
printf("Enter %d numbers: ", len);
for (i = 0; i < len; i++)
   scanf("%d", &array[i]);
```

Some Notes about VLAs

- Exercise: rewrite the binary search program using a VLA so that the user enters array length first
- VLAs can be multidimensional, but cannot have initializers; i.e., we have to assign them explicitly

Example: `latin.c`

- $n \times n$ matrix is a Latin square if
  - each row is a permutation of $\{1, 2, \ldots, n\}$
  - each column is a permutation of $\{1, 2, \ldots, n\}$
- Examples:

```
1 2 3 is Latin      1 2 3 is not Latin
2 3 1 square        3 1 2 square
3 1 2               1 3 2
```

- Completed Sudoku puzzles are 9×9 Latin squares with some additional constraints

Program `latin.c`

```
/* Program latin.c */
#include <stdio.h>

int main() {
  int size = 0, i, j;

  printf("Enter the size of the latin square: ");
  scanf("%d", &size);
  if (size < 1) return 1;

  int square[size][size], occurs[size+1];

  printf("Enter a matrix (%d by %d) of integers in "
         "the range [1-%d]: \n", size, size, size);
```

`latin.c`: Reading the matrix

```
for (i = 0; i < size; i++)
  for (j = 0; j < size; j++) {
    scanf("%d", ________________ );

    if (square[i][j] > size || square[i][j] <= 0) {
      printf("Error: element out of range.\n");
      return 1;
    }
  }
```

`latin.c`: Checking Rows

```
for (i = 0; i < size; i++) {
  for (j = 1; j <= size; j++)
    occurs[j] = 0;

  for (j = 0; j < size; j++) {
    if ( ____________________ > 0) {
      printf("This is not a Latin square.\n");
      return 1;
    }
    else
      occurs[ _________________ ] = 1;
  }
}
```

`latin.c`: Checking Columns

```
  for (j = 0; j < size; j++) {
    for (i = 1; i <= size; i++) occurs[i] = 0;

    for (i = 0; i < size; i++) {
      if (_____________________________ > 0) {
        printf("This is not a Latin square.\n");
        return 1;
      }
      else occurs[ _______________ ] = 1;
    }
  }

  printf("This is a Latin square.\n");
  return 0;
}
```