

# EECS 22: Advanced C Programming

## Lecture 5

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering  
Electrical Engineering and Computer Science  
University of California, Irvine

## Lecture 5: Overview

- Review of the C Programming Language
  - Introduction to Data Structures
  - Arrays
    - Introduction
    - Indexing
    - Initialization
  - Multi-Dimensional Arrays
  - Program Example `Histogram.c`

## Review of the C Programming Language

- Introduction to Data Structures
  - Until now, we have used only single data elements of basic (non-composite) type
    - integral types
    - floating point types
  - Most programs, however, require complex *data structures* using composite types
    - arrays, lists, queues, stacks
    - trees, graphs
    - dictionaries
  - ANSI C provides built-in support for
    - arrays
    - structures, unions, enumerators
    - pointers

EECS22: Advanced C Programming, Lecture 5

(c) 2016 R. Doemer

3

## Arrays

- Array data type in C
  - Composite data type
    - Type is an array of a sub-type (e.g. array of `int`)
  - Fixed number of elements
    - Array size is fixed at time of definition (e.g. 100 elements)
  - Element access by index (aka. subscript)
    - Element-access operator: `array[index]` (e.g. `A[42]`)
- Example:

```
int A[10]; /* array of ten integers */  
  
A[0] = 42; /* access to elements */  
A[1] = 100;  
A[2] = A[0] + 5 * A[1];
```

EECS22: Advanced C Programming, Lecture 5

(c) 2016 R. Doemer

4

## Arrays

- Array Indexing
  - Start counting from 0
    - First element has index 0
    - Last element has index  $Size-1$

- Example:

```
int A[10];

A[0] = 42;
A[1] = 100;
A[2] = A[0] + 5 * A[1];
A[3] = -1;
A[4] = 44;
A[5] = 55;
/* ... */
A[9] = 99;
```

	A
0	42
1	100
2	542
3	-1
4	44
5	55
6	0
7	0
8	0
9	99

EECS22: Advanced C Programming, Lecture 5

(c) 2016 R. Doemer

5

## Arrays

- Array Indexing
  - for loops are often very helpful
    - `for(i=0; i<N; i++)`  
`{...A[i]...}`

- Example:

```
int A[10];
int i;

for(i=0; i<10; i++)
{ A[i] = i*10 + i;
}

for(i=0; i<10; i++)
{ printf("%d, ", A[i]);
}
```

	A
0	0
1	11
2	22
3	33
4	44
5	55
6	66
7	77
8	88
9	99

```
0, 11, 22, 33, 44, 55, 66, 77, 88, 99,
```

EECS22: Advanced C Programming, Lecture 5

(c) 2016 R. Doemer

6

## Arrays

- Array Indexing
  - Array indices are *not* checked by the compiler, nor at runtime!
  - Accessing an array with an *index out of range* results in undefined behavior!
- Example:
 

```
int A[10];
int i;

A[-1] = 42; /* INVALID ACCESS! */

for(i=0; i<=10; i++)
  /* INVALID LOOP RANGE! */
  { printf("%d, ", A[i]);
  }
```

0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

EECS22: Advanced C Programming, Lecture 5 (c) 2016 R. Doemer 7

## Arrays

- Array Initialization
  - Static initialization at time of array definition
  - Initial elements listed in { }
- Example:
 

```
int A[10] = { 42, 100,
              310, 44,
              55, 0,
              3, 4,
              0, 99};
```

	A
0	42
1	100
2	310
3	44
4	55
5	0
6	3
7	4
8	0
9	99

EECS22: Advanced C Programming, Lecture 5 (c) 2016 R. Doemer 8

## Arrays

- Array Initialization
  - Static initialization at time of array definition
  - Initial elements listed in { }

- Example:

```
int A[ ] = { 42, 100,
            310, 44,
            55, 0,
            3, 4,
            0, 99};
```

- With given initializer list, array size may be omitted
  - automatically determined

	A
0	42
1	100
2	310
3	44
4	55
5	0
6	3
7	4
8	0
9	99

## Arrays

- Array Initialization
  - Static initialization at time of array definition
  - Initial elements listed in { }

- Example:

```
int A[10] = { 1, 2, 3};
```

- With given initializer list *and* array size, unlisted elements are zero-initialized
  - array is filled up with zeros

	A
0	1
1	2
2	3
3	0
4	0
5	0
6	0
7	0
8	0
9	0

## Multi-Dimensional Arrays

- Multi-dimensional arrays are modeled as *Arrays of arrays (of arrays...)*
- Example:

```
int M[3][2] = {{1, 2},
              {3, 4},
              {5, 6}};

int i, j;

for(i=0; i<3; i++)
  { for(j=0; j<2; j++)
    { printf("%d ",
            M[i][j]);
      }
    printf("\n");
  }
```

M	0	1
0	1	2
1	3	4
2	5	6

```
1 2
3 4
5 6
```

EECS22: Advanced C Programming, Lecture 5

(c) 2016 R. Doemer

11

## Multi-Dimensional Arrays

- Storage Allocation for Multi-Dimensional Arrays
  - Example: `Array[Dim1][Dim2]...[DimN]`
    - Need space for  $Dim1 * Dim2 * \dots * DimN$  elements
      - `sizeof(int[5][100]) = sizeof(int)*5*100`
- Storage in Linear Address Space in Memory
  - In storage order, right-most index varies the fastest!
  - To obtain the linear offset of a given array element, left indices need to be multiplied by the sizes on the right
    - `int A[10];`      `A[5]` is the 5<sup>th</sup> element (starting with 0<sup>th</sup> element)
    - `int M[6][7];`      `M[3][2]` is the  $3*7+2 = 23^{\text{rd}}$  element
    - `int T[2][8][5];`    `T[1][2][3]` is the  $1*8*5+2*5+3 = 53^{\text{rd}}$  element
  - When declaring arrays as parameters for functions, the left-most size is irrelevant and can be omitted
    - `int f(int M[][7]);`

EECS22: Advanced C Programming, Lecture 5

(c) 2016 R. Doemer

12

## Multi-Dimensional Arrays

- Example: Allocation of 2-dimensional Arrays: Tables
  - Rows (major)
  - Columns (minor)

```
#define ROWS 3
#define COLS 2
int M[ROWS][COLS]
= {{1, 2},
   {3, 4},
   {5, 6}};
```

M	0	1
0	1	2
1	3	4
2	5	6

- Linear memory layout:

```
int M[ROWS][COLS] = {{1, 2}, {3, 4}, {5, 6}};
```

```
M[0][0] M[0][1] M[1][0] M[1][1] M[2][0] M[2][1]
```

1	2	3	4	5	6
---	---	---	---	---	---

## Program Example

- **Histogram.c**
  - Display a simple bar chart for 10 integer values
- Desired output:

```
% Histogram
Please enter data value 1: 111
Please enter data value 2: 222
Please enter data value 3: 33
Please enter data value 4: 333
[...]
Please enter data value 10: 111
1: 111 *****
2: 222 *****
3: 33 ****
4: 333 *****
[...]
10: 111 *****
%
```

## Program Example

- **Histogram.c** (part 1/3)

```
/* Histogram.c: print a histogram of data values */
/* author: Rainer Doemer */
/* modifications: */
/* 11/02/04 RD initial version */

#include <stdio.h>

/* constants */
#define NUM_ROWS 10

/* main function */
int main(void)
{
    /* variable definitions */
    int Data[NUM_ROWS];
    int i, j, max;
    double scale;

    ...
}
```

## Program Example

- **Histogram.c** (part 2/3)

```
...
/* input section */
for(i = 0; i < NUM_ROWS; i++)
{ printf("Please enter data value %2d: ", i+1);
  scanf("%d", &Data[i]);
} /* rof */

/* computation section */
max = 0;
for(i = 0; i < NUM_ROWS; i++)
{ if (Data[i] > max)
  { max = Data[i];
    } /* fi */
} /* rof */
scale = 70.0 / max;

...
}
```



## Program Example

- **Histogram.c** (part 3/3)

```

...
/* output section */
for(i = 0; i < NUM_ROWS; i++)
{ printf("%2d: %5d ", i+1, Data[i]);
  for(j = 0; j < Data[i]*scale; j++)
  { printf("*");
    } /* rof */
  printf("\n");
} /* rof */

/* exit */
return 0;
} /* end of main */

/* EOF */

```

## Program Example

- **Example session: Histogram.c**

```

% vi Histogram.c
% gcc Histogram.c -o Histogram -Wall -ansi
% Histogram
Please enter data value 1: 11
Please enter data value 2: 22
Please enter data value 3: 3
Please enter data value 4: 33
Please enter data value 5: 44
Please enter data value 6: 55
Please enter data value 7: 66
Please enter data value 8: 33
Please enter data value 9: 22
Please enter data value 10: 22
1: 11 *****
2: 22 *****
3: 3 ****
4: 33 *****
5: 44 *****
6: 55 *****
7: 66 *****
8: 33 *****
9: 22 *****
10: 22 *****
%

```