Programming in C

Chapter - 5

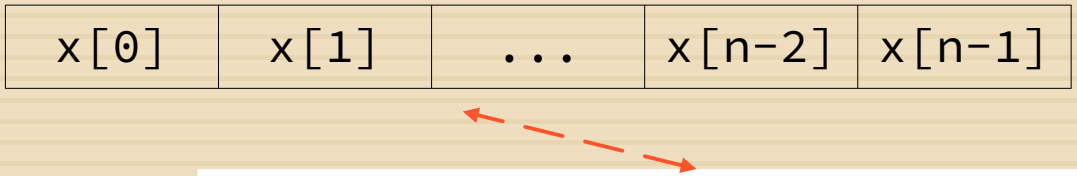
Arrays and Strings

by, Santa Basnet

Arrays

introduction

- In C language, Arrays are collection of similar type of data elements.
- Stored in contiguous memory location.
- Data elements in an array are accessed through one or more integer indices, called subscripts, enclosed in square brackets i.e. [].
- Array index always starts from 0.



x is an n-element, one dimensional array.

Arrays

initialization

- Array definition & assignment:

```
/* Array Declaration */
/* storage_class type array_name[array_size]; */
/* Example: */

 int digits[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

 char color[3]  = {‘R’, ‘G’, ‘B’}

 static float x[6] = {0, 0.25, 0, -0.50f, 0, 0}
```

Arrays

initialization

- Array definition & assignment:

```
/* Array Declaration */
/* storage_class type array_name[array_size]; */
/* Example: */

 int digits[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

 char color[3]  = {'R', 'G', 'B'}

 static float x[6] = {0, 0.25, 0, -0.50f, 0, 0}
```

```
What happens if we do this ?

printf("%c\n", color[4]); /* Undefined Behavior */
```

Arrays

initialization

- Processing an Array:

```
/* Array Declaration */
/* storage_class type array_name[array_size]; */
/* Example: */

 int digits[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
 for(int i = 0; i < 10; i++){
        printf("%d ", digits[i]);
 }
 // ??

 char color[4]  = {'R', 'G', 'B', '\0'};
 printf("%s\n", color);
 // ??
```

Arrays

Multi-Dimensions

- Multi-dimensional Arrays:

```
int tic_tac_toe[3][3];
```

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]

Arrays

Multi-Dimensions

- Multi-dimensional Arrays: Assignment

```
int tic_tac_toe[3][3] = {
                            {0, 1, 2},
                            {3, 4, 5},
                            {6, 7, 8}
                        };
```

[0][0] = 0	[0][1] = 1	[0][2] = 2
[1][0] = 3	[1][1] = 4	[1][2] = 5
[2][0] = 6	[2][1] = 7	[2][2] = 8

Arrays

Multi-Dimensions

- Multi-dimensional Arrays: Assignment

```
int tic_tac_toe[3][3] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
```

[0][0] = 0	[0][1] = 1	[0][2] = 2
[1][0] = 3	[1][1] = 4	[1][2] = 5
[2][0] = 6	[2][1] = 7	[2][2] = 8

Arrays

Multi-Dimensions

- Multi-dimensional Arrays: Assignment

```
int chess_knight[8][8] = ??

      How do you assign it ?
```

Arrays

Operations on Arrays

- Processing an Array:

```
int mat_1[3][3] = {
                      {0, 1, 2},
                      {3, 4, 5},
                      {6, 7, 8}
                   };
int mat_2[3][3] = {
                      {8, 7, 6},
                      {5, 4, 3},
                      {2, 1, 0}
                   };
int result[3][3]; // How to add two matrices ?
```

Arrays

Operations on Arrays

- Processing an Array:

Other array operations ???

```
int first_ten[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

int result[10] = {0, 1, 3, 27, 64, 125, 216, 343,
                  512, 729};
```

How do you achieve this ??

Arrays

Multi-dimensions

- Higher dimensions:

```
int arr[10][20][30] = {
                        {
                                {1, 2, 3, 4},
                                {5, 6, 7, 8},
                                {9, 10, 11, 12},
                        },
                        {
                                {21, 22, 23, 24},
                                {25, 26, 27, 28},
                                {29, 30, 31, 32},
                        }
                };
```

Arrays

Strings

- String representation:
- Are characters array with terminated by ‘\0’ i.e. (a last character)

```
/* Example: */

 char name[20] = {‘E’,‘V’,‘E’,‘R’,‘E’,‘S’,‘T’,‘\0’};
```

```
/* Example: */

 const char name[20] = “EVEREST”;
```

Arrays

Strings

- String representation:
- Are characters array with terminated by ‘\0’ i.e. (a last character)

```
/* Example: */

 char name[20] = {‘E’,‘V’,‘E’,‘R’,‘E’,‘S’,‘T’,‘\0’};
```

```
/* Example: */

 const char name[20] = “EVEREST”;
```

Arrays

Strings

- There is no String type in C!
- Strings are implemented as a character array, name[] or *name.
- String literals are enclosed by double quotes, “Hello”.
- Terminated by NULL character, ‘\0’.
- Format specifier: “%s”.
- Same as:

```
char text[] ={'H','e','l','l','o','\0'}
```

Arrays

Strings

- String representation:
- Name lists:

```
/* Example: */

 char name[3][10] = {“RAM”, “SHYAM”, “HARI”};
```

Arrays

Strings

- String representation:
- Name lists:
- Utility functions: <string.h>

```
/* Example: */

 char name[3][10] = {“RAM”, “SHYAM”, “HARI”};

 int len = strlen(name[0]);
 char name1[32];
 strcpy(name1, name[0]);
 strcat(name1, name[1]);
 strcmp(name1, name[1]);
```

Thank you.

Questions ?